POLITECNICO DI TORINO

MASTER'S DEGREE IN COMMUNICATIONS AND COMPUTER NETWORKS ENGINEERING

Master's Degree Thesis

Techniques for attitude determination using GNSS carrier phase observations

---

Candidate:
Vincenzo CENTRONE

Supervisor:
Prof. Fabio DOVIS
Company supervisor:
Daniel ARIAS MEDINA
(DLR)

April, 2019

A thesis submitted in fulfillment of the requirements for the Master's degree in
Communications and Computer Networks Engineering,

written in and financed by the

Nautical Systems department of the **Institute of Communication and Navigation**

of the **German Aerospace Center (DLR)**

POLITECNICO DI TORINO

Abstract

Techniques for attitude determination using GNSS carrier phase observations

by Vincenzo CENTRONE

Global Navigation Satellite Systems(GNSS) are the cornerstone and main information supplier for PNT (Positioning, Navigation and Timing) information. With the growing interest towards autonomous vehicles, it is of great importance to guarantee the accuracy and reliability of navigation estimates provided by GNSS. Other than providing the absolute localization and timing, GNSS signals can be also used to compute the orientation of a platform when multiple antennas are used. Attitude determination is the process of estimating the orientation of a rigid body with respect to its environment and it constitutes a fundamental task for the navigation of spacecraft and other vehicles of large inertia. GNSS represents an appealing alternative for attitude determination. Employing a setup of multiple GNSS antennas rigidly mounted on a vehicle, one seeks to find the rotation which relates the local and global navigation frames. Accurate attitude estimation based on GNSS requires the use of GNSS carrier phase observations. In depth examination on the direct estimation of the attitude has not been yet addressed. Among the wide variety of attitude parametrizations, quaternion rotation has become the standard and most widely applied rotation characterization for robotic or computer vision applications. This knowledge or expertise has not been fully exploited for navigation purposes, or at least for GNSS carrier phase-based attitude determination. This work explores the possibilities of this approach and investigates how the Ambiguity Resolution procedure can be enriched with a constrained attitude problem already during the phase of float estimation.

Keywords: *Attitude determination; Quaternion; LAMBDA method; Carrier phase observations; GNSS*

Acknowledgements

This thesis work took place as part of a part time working experience at the ***department of Nautical Systems***, which is part of the ***Institute of Communication and Navigation*** of the ***German Aerospace Center (DLR)***, in Germany. There are people that helped and supported me during this thesis work and my Master's degree career. First of all, I want to thank my supervisor at the Politecnico di Torino Prof. Fabio Dovis for introducing me to this unique opportunity and for his precious classes on GNSS systems. Then, I want to thank my supervisor at DLR, Daniel Arias Medina, for introducing me to this new topic and supporting me during my permanence in DLR. Furthermore, I also want to express my gratitude to all the people who worked with me and helped me with their knowledge and valuable comments. In addition, special thanks go to Caro, Jack, Liang and Yuliia for the nice time spent together and their support during my stay in Neustrelitz. Last, but not least, I also want to thank Claudia, Bj, Federico, Carles, all my colleagues in Torino, as well as, my friends and my family for their support during my whole Master's degree career.

Contents

List of Figures

List of Tables

List of Acronyms

GNSS	**G**lobal **N**avigation **S**atellite **S**ystem
ECEF	**E**arth **C**entered **E**arth **F**ixed
TEC	**T**otal **E**lectron **C**ontent
DOP	**D**ilution **O**f **P**recision
PDOP	**P**osition **D**ilution **O**f **P**recision
LAMBDA	**L**east (squares) **Amb**iguity **D**ecorrelation **A**djustment
QUEST	**Q**uaternion **Est**imator
SPP	**S**ingle **P**oint **P**ositioning
PPP	**P**recise **P**oint **P**ositioning
RTK	**R**eal **T**ime **K**inematic
SD	**S**ingle **D**ifference
DD	**D**ouble **D**ifference
LS	**L**east **S**quares
ILS	**I**nteger **L**east **S**quares
AR	**A**mbiguity **R**esolution

Chapter 1

Introduction

Global Navigation Satellite Systems(GNSS) are the cornerstone and main information supplier for Positioning, Navigation and Timing (PNT) information. With the growing interest towards autonomous vehicles, it is of great importance to guarantee the accuracy and reliability of navigation estimates provided by Global Navigation Satellite Systems (GNSS). Other than providing the absolute localization and timing, GNSS signals can be used to compute the orientation of a platform when multiple antennas are used. Attitude determination is the process of estimating the orientation of a rigid body with respect to its environment and it constitutes a fundamental task for the navigation of spacecraft and other vehicles of large inertia. GNSS represents an appealing alternative for attitude determination, providing a drift-less absolute orientation solution while posing minimal requirements in terms of cost, weight and power consumption. Employing a setup of multiple GNSS antennas rigidly mounted on a vehicle, one seeks to find the rotation which relates the local and global navigation frames. Accurate attitude estimation based on GNSS requires the use of GNSS carrier phase observations.

On their series of work (like [7] [5]), Teunissen and Giorgi introduced the Multivariate-Constrained LAMBDA (MC-LAMBDA) for carrier phase-based attitude determination. MC-LAMBDA modifies the traditional process of integer search to also account for the nonlinear constraints that follow from knowing a-priori the relative positions of the antennas in a local frame. However, in depth examination on the direct estimation of the attitude has not been yet addressed. Among the wide variety of attitude parametrizations, quaternion rotation has become the standard and most widely applied rotation characterization for robotic or computer vision applications. This knowledge or expertise has not been fully exploited for navigation purposes, or at least for GNSS carrier phase-based attitude determination. This work explores the possibilities of this approach and investigates how the Ambiguity Resolution procedure can be enriched with a

constrained attitude problem already during the phase of float estimation.

1.1 Objectives

The aim of this thesis is to study attitude problem, first from a general point of view and, then, for in GNSS. Moreover, it has the goal of comparing the results, in terms of success ratio, between the classical LAMBDA method and the Quaternion based one.
This work is composed by the following tasks:

1. Study and review the methods for attitude determination using baseline observations, applying a variety of methods
2. Getting the basis of GNSS-based positioning and attitude approaches
3. Implementation of quaternion based algorithm
4. Evaluation

The development of this work has been financed and supported by the ***department of Nautical Systems***, which is part of the ***Institute of Communication and Navigation*** of the ***German Aerospace Center (DLR)*** for a period of six months. The initial schedule established in order to fulfil all the proposed tasks evolved as new techniques were investigated and some other were considered as a dead-end.

All the algorithms have been built in MATLAB, as it is convenient for fast prototyping, as well as presenting several advantages such as its large database of statistical functions, the capability to work with matrices or its appealing debugging process.

1.2 Thesis outline

The rest of this work is organized as follows:
In the Chapter 2, a brief introduction on GNSS is provided. This is done only to introduce the terms and the notation, related to GNSS, which are used and mentioned in the thesis. It is not meant to be an exhaustive explanation of the working principle of GNSS.
After that, the Chapter 3, takes into account account what is the attitude problem. At the beginning, a definition of the attitude problem and why it is important. Then, the most famous algorithms developed to solve it and

how this problem can be related to the GNSS. It means, how to use the informations on the antenna positions to estimate the orientation of a rigid body and what parameters can affect the accuracy of the results.
Following, in the Chapter 4, the GNSS attitude problem is faced together with the issues derived by the usage of the phase measurements, which are more precise. Thus, the concept of single and double differences is introduced, as well as, a linearized model for them, using the unit line of sight vectors. Then, the classical approach (LAMBDA method) is analysed, described and compared to a new approach. The latter has been called *Quaternion approach*.
Then, in the last Chapter, the simulations and the final results are displayed and commented. This Chapter is divided in three parts. The first two of them refers to the theory explained in the third Chapter, i.e. to the algorithm to solve the attitude and on the parameters affecting the accuracy on the estimated Euler angles. On the contrary, the last part is related to the fourth Chpater. So, the two different approaches for the actual GNSS attitude problem are compared and the effect of some parameters, like the accuracy of the code measurements and the baseline length, is studied.
Finally, a brief Conclusion, with the next steps to follow in this research topic is reported.

Chapter 2

GNSS

In this first Chapter a small introduction on the main idea of GNSS is presented. It is meant to give only an introduction to the terminology and concepts used in the following chapters, but without the usage of many mathematical solutions and equations. For a more detailed explanation it is suggest to read one (or more) among the books [10] [31] [12] [20] of the bibliography.

First of all, GNSS stands for Global Navigation Satellite Systems. It relies on a constellation of satellites, which follow an elliptical orbit (with small eccentricity) around the Earth, broadcasting signals in direction of the Earth surface, with the goal of providing a full coverage.

FIGURE 2.1: Example of satellites constellation to provide a full coverage [34].

There are different system constellations available at the moment. The most important ones are surely [31]:

- GPS - United States of America

- GLONASS - Russia
- Galileo - European Union
- Beidou - China

Using these signals is, then, possible to estimate the position, velocity and time (PVT) of a user. How is it possible?

2.1 Reference Coordinate Systems

First of all, a position has some meaning only if it is referred to a reference. For this reason, is very important to present what are the existing coordinate systems. Moreover, their relation has to be understood, because GNSS measures the position of the receiver with respect to a constellation of satellites, but the inertial sensors, like the gyroscopes, take measurements related to a body frame. Here the main coordinate systems are presented.

Geodetic Coordinate System. To characterize the position of a user three coordinates are necessary and in this case they are defined as latitude (φ), longitude (λ) and height (h). See Figure 2.2. The longitude is defined as the angle between the Prime meridian and the point in which we have the user position. The latitude is the angle between the equatorial plane of the ellipsoid and the normal to the surface, passing for the measured point. To conclude, the height is the distance, measured over the perpendicular line to the ellipsoid, between the measured point and the ellipsoid itself [20].

Earth-Centered Inertial Coordinate System. It is useful for the representation of the orbits of the satellites. In fact, in the ECI frame the origin is at the center of gravity of the Earth and the axes are fixed and defined with respect to the stars, creating an orthogonal coordinate system [12]. See Figure 2.2.

Earth-Centered Earth-Fixed Coordinate System. The idea is really similar to the ECI frame. The difference is that for the ECEF frame all the axes are fixed with respect to the Earth and not to the "sky". As can be seen in Figure 2.2, the x axis passes through the intersection of the Greenwich meridian (defining the 0° longitude) with the equatorial plane. The z axis points to the North Pole and the y axis forms an orthogonal frame with the other two axes and it is in the equatorial plane [9]. The ECEF it is used, usually, as reference frame (in particular in this thesis work).

Local North-East-Down Coordinate System. This is known as local navigation frame and it is fixed with respect to the Earth surface. In fact, the

origin (denoted as O_n in Figure 2.2) is an arbitrary point on the the Earth (generally the user position or the centre of gravity of the vehicle). The x and y axes point toward the geodetic north and east directions, whereas the z axis points along the normal to the ellipsoid, in the downward direction [9].

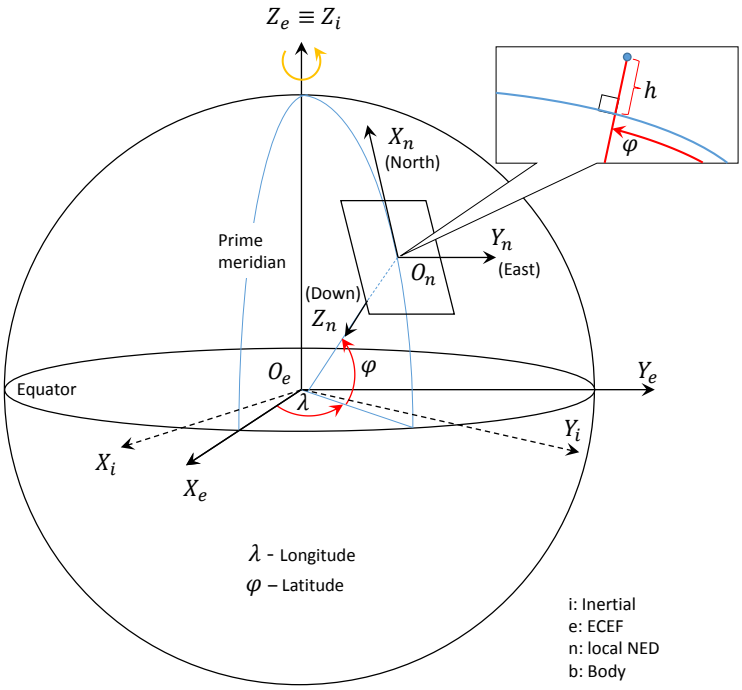

FIGURE 2.2: Coordinate systems: Geodetic $\{\lambda, \varphi, h\}$, ECI $\{X_i, Y_i, Z_i\}$, ECEF $\{X_e, Y_e, Z_e\}$ and NED $\{X_n, Y_n, Z_n\}$ represented [2].

Body Coordinate System. It is also known as vehicle frame. The origin is the same as the local NED frame, but the axis do not depend on the position on the Earth surface, but they remains fixed with respect to the vehicle. They are usually defined as in Figure 2.3 and they are also called roll, pitch and yaw (or heading) axes [9]. However, in reality, the position and the axis of the sensors are not perfectly aligned with the rest of the body. For this reason, the body of the vehicle and the body of the measured sensors are not exactly the same.

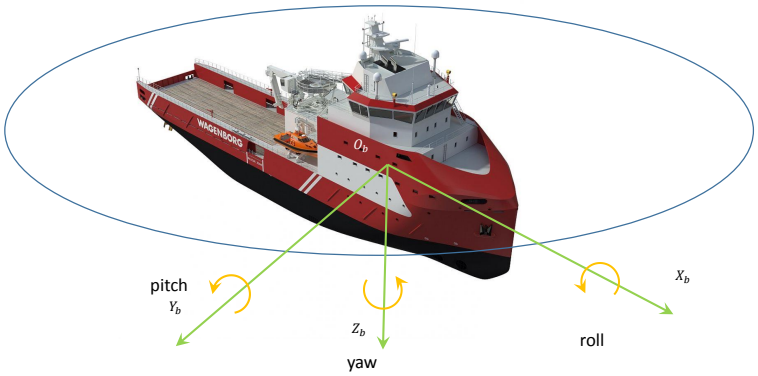

FIGURE 2.3: An example of body coordinate system [2].

2.2 Estimating the position

Now, how is it possible to estimate the position of a body using GNSS? The idea is really simple. In fact, the position of satellites are known and they all broadcast a signal. Moreover, their clocks are synchronous. So, the user can measure the time needed to receive it. It means that the TOA (time of arrival) is utilized. In this way, to each measurement a sphere can be associated by multiplying the time with the speed of light in order to obtain the distance between the user and satellite:

$$\rho = c\ \tau_j \tag{2.1}$$

where τ_j is the measured time of arrival from the satellite j, R_j is the estimated distance (*pseudorange*) and c the speed of light.

Finally, by collecting measurements from different satellites is possible to discover the position as intersection of these spheres (spherical positioning), as represented in Figure 2.4.

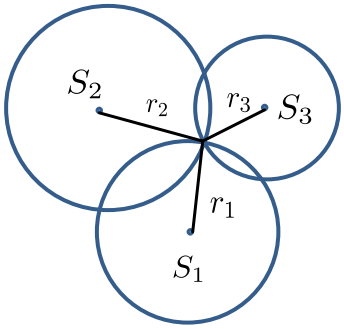

FIGURE 2.4: Spherical positioning concept in a 2D case.

It can seem that in a 3D case to find a solution at least 3 spheres, and so 3 satellites in view are needed, because the only unknowns are the coordinates of the user (x_u, y_u, z_u) in the reference frame. However, in a real scenario at least 4 satellites are needed. In fact, the user is not synchronized with the transmitters. This leads to a clock offset, which has to be estimated, and, consequently, to a new unknown (δt_u).

$$\rho_j = c\ \tau_j + c\ \delta t_u \tag{2.2}$$

Through a linearization process, using the LOS vectors u_j (linking the user to the satellite j) the problem can be easily solved in a fast iterative way [10].

2.3 Sources of errors and DOP

Actually, it is not all so easy. Indeed, the pseudorange measurements in a real scenario are affected by many kinds of error. They are [10]:

- *Ionospheric delay.* The propagation speed through the ionosphere (a part of the atmosphere) depends on the frequency of the signal and on the TEC (total electron content). In particular, for higher frequencies the delay is lower. However, the difficult part is to estimate the TEC, which depends on the Sun radiation.
- *Tropospheric delay.* in few words, it depends on the weather conditions (wet or dry, pressure, etc.).

- *Relativistic effects.* Caused by the change of the gravitational potential and the eccentricity of the satellite's orbit. Due to these factors, the clocks on board clocks and the ones on the Earth differs.
- *Multipath.* Reception of delayed replicas of the same signal, which is reflected by obstacles.
- *Noise at the receiver and instrumental delays.*
- *Errors regarding the control systems*. For example, errors on ephemeris, for code generations, etc.

Each one of these error is modelled as a Gaussian noise with zero mean. In fact, all the biases of are estimated and, in some way, removed to avoid accuracy problems. In fact, it would be an issue to have solution with high precision but low accuracy. Moreover, all the errors are considered to be independent. As a consequence the total error contribution, which is called UERE (User equivalent range error), is also modelled as a Gaussian random variable with zero mean and variance σ^2_{UERE}, given by the combination of all the single variances.

By the way, there is another factor which can cause a worsening of the performance of GNSS. This is a geometrical factor, called DOP (Dilution of Precision). In fact, it is also really important how the satellites in view are distributed in the sky. A graphical representation of this problem is present in Figure 2.6.

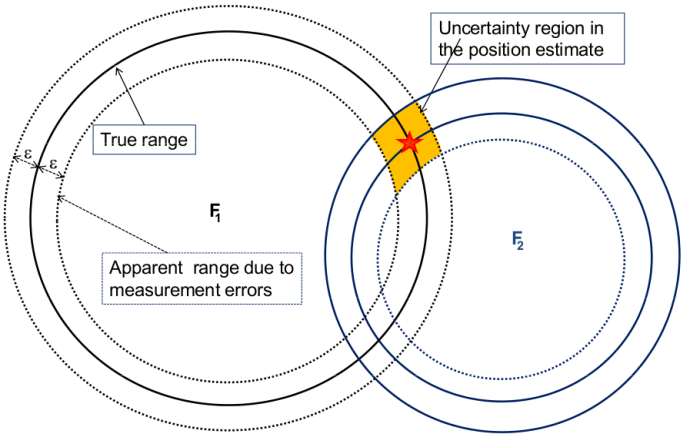

FIGURE 2.5: Representation of the uncertainty region for the position. ϵ represents the measurements error [10].

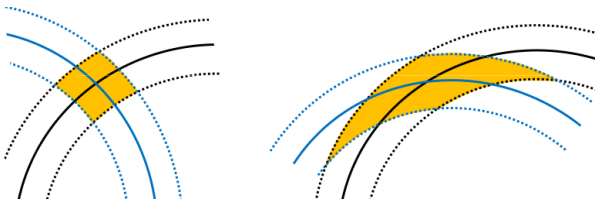

FIGURE 2.6: Effect of the displacement of the satellites in the sky (DOP) on the uncertainty region of the position [10]. On the right a worse DOP is obtained and, so, the uncertainty region is larger.

2.4 Code and phase measurements

So, from what said before, the final model for the pseudorange is :

$$\rho_k^i = \left\| p^i - p_k \right\| + c\delta t_k - c\delta t^i + Tr^i + I_0^i + \varepsilon_k^i \tag{2.3}$$

where p^i and p_k are the position of the satellite i and the receiver k. δt_k and δt^i are the clock errors, Tr^i is the tropospheric delay, I_0^i the ionospheric delay and, finally, ε_k^i is a noise term which takes into account multipath effects, instrumental delays, phase biases, etc.
By the way, what has been described until now is only the code measurement. In fact, in GNSS there are 2 kinds of measurements:

- *code measurements*. In this case is measured the difference, in *time* (Δt), between the code received by the transmitter and a replica generated, locally, at the receiver. $\rho = c\Delta t$
- *phase measurements*. It is found the difference, in phase (θ), between a local carrier and the one received by the satellite. However, since the carrier wavelength (λ) is very small, is also important to estimate the number of times in which the carrier has been repeated (a), while travelling in direction of the receiver. $\Phi = (a + \theta)\lambda$

The model for the phase measurement, which is given in meters, is:

$$\Phi_k^i = \left\| p^i - p_k \right\| + c\delta t_k - c\delta t^i + Tr^i - I_0^i + \lambda a_k^i + \varepsilon_k^i \tag{2.4}$$

where λ is the wavelength of the signal, which is 19cm for GPS L1, and N is the ambiguity. These are the only new terms with respect to the code observations.

To conclude, the main difference, for what concerns this thesis work, can be described as follow: code measurements are noisy but unambiguous, whereas phase observations are precise but affected by ambiguity. It means that phase observations are characterized by a noise which is two orders of magnitude lower than code pseudorange observations, but are ambiguous by an unknown number of integer ambiguities [10].

Chapter 3

Attitude determination

The position is not the only thing we could be interested in. For many application is also very important to know the orientation of an object (body). This is not an easy task and this issue is known as *attitude determination problem*. The goal of attitude determination is, in general, to estimate the orientation of a moving body (spacecrafts, vessels, airplanes, etc.), with respect to a reference frame or some specific object of interest. To achieve this, the rotation which carries a set of reference unit vectors into a set of observation unit vectors has to be found (see Figure 3.1). Attitude determination has been a recurrent problem in spacecraft systems, where the two vectors are typically the unit vector to the Sun and the Earth's magnetic field vector for coarse "sun-mag" attitude determination or unit vectors to two stars tracked by two star trackers for fine attitude determination [15] [37].

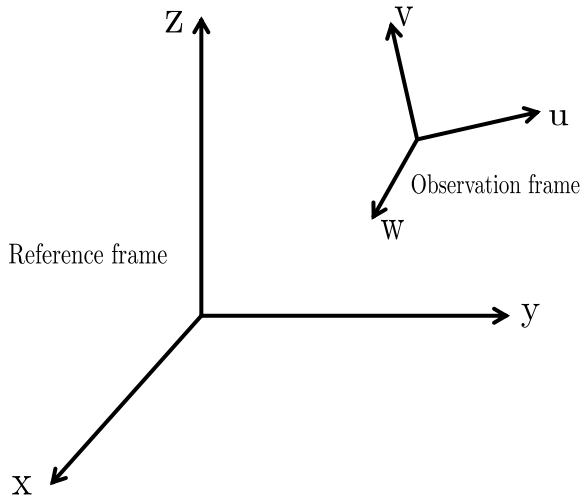

FIGURE 3.1: Reference and body frame example. The goal of attitude determination is to estimate the rotation relating the two frames.

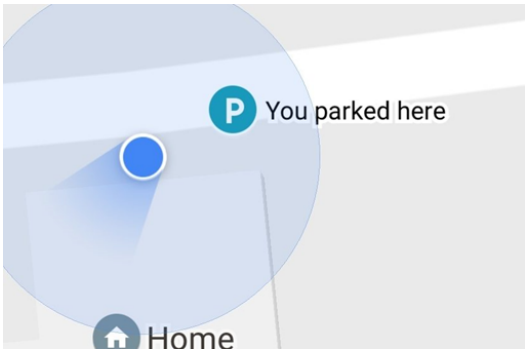

FIGURE 3.2: Practical example of the importance of attitude determination [8].

But, first of all, why could it be important to determine the attitude? Some examples could be the following ones: to point solar panels (for power generation), to point directional antennas, to orient spacecraft for orbit maneuvers, to orient vessels or airplanes for maneuvers and to reduce the risk of collisions.

3.1 Attitude Parametrization

Euler angles

So, the simplest approach would be, from a very intuitive point of view, to find the Euler angles defining the rotation from the local frame to the reference one. In fact, a rotation can be seen as 3 consecutive rotations applied over the three axis of the body coordinate system. The three angles, as showed in Figure 3.3, are called Roll (φ), Pitch (θ) and Heading (ψ) and they are respectively the rotations around the longitudinal, transverse and vertical axis. Although, the usage of the Euler angles is really easy to understand, they suffer from the so called gimbal lock, i.e. a singularity problem when the pitch angle is close to the $\pm 90^{\circ}$ value, since the other two axis get aligned.

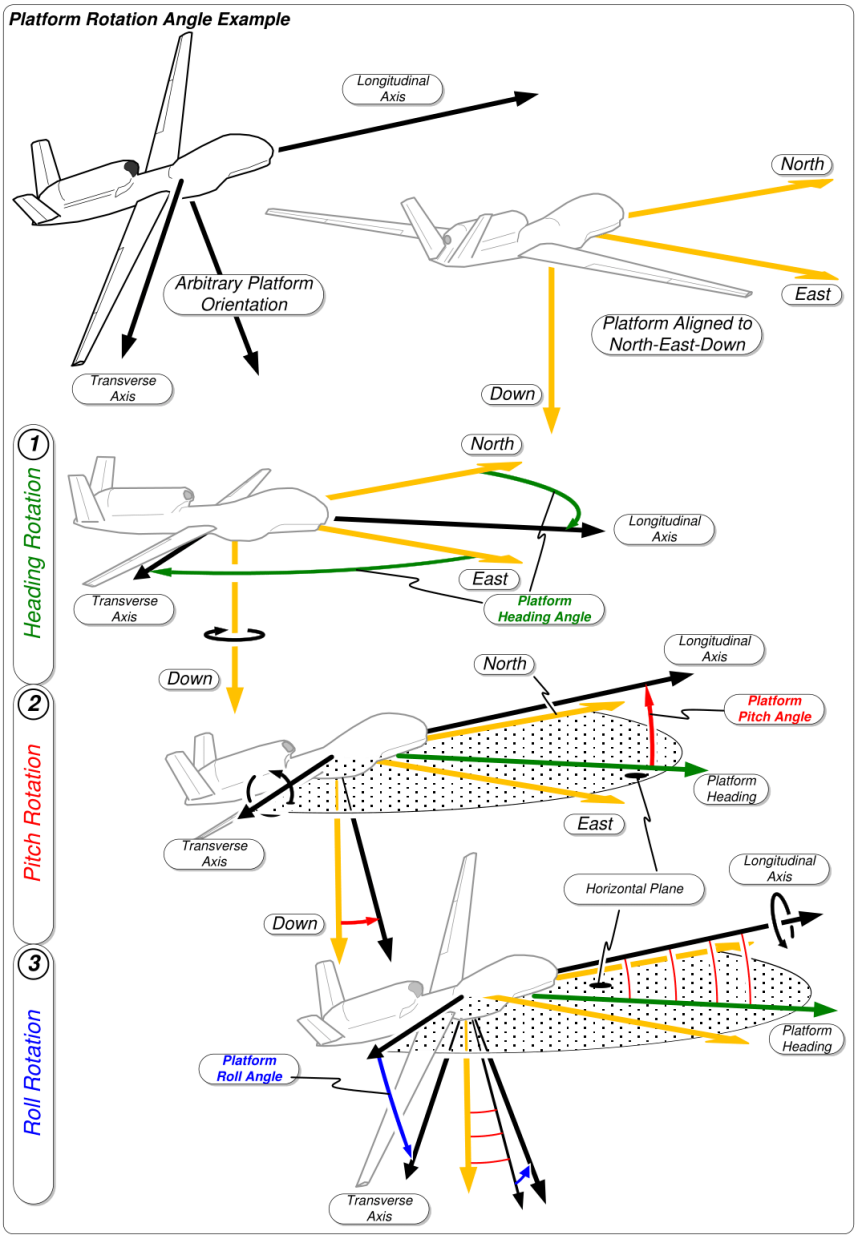

FIGURE 3.3: Representation of the Euler angles together with the axis of rotation [35].

Rotation matrix

This problem of the Euler angles is solved using a representation which is not affected by the singularities, i.e. the rotation matrix R. It is a 3×3 matrix, i.e. it contains nine elements, representing a 3D rotation. It can be

simply obtained as product of the single rotations defined by the 3 Euler angles, as shown in the following equations.

$$R = R_{123}(\varphi, \theta, \psi) = R_1(\varphi)R_2(\theta)R_3(\psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\varphi s_\theta c_\psi - c_\varphi s_\psi & s_\varphi s_\theta s_\psi + c_\theta c_\psi & c_\theta s_\varphi \\ c_\varphi s_\theta c_\psi + s_\varphi s_\psi & c_\varphi s_\theta s_\psi - s_\varphi c_\psi & c_\theta c_\varphi \end{bmatrix} \tag{3.1}$$

where c_θ, c_φ, c_ψ, s_θ, s_φ and s_ψ are the cosine and sine of the angles:

$$c_\varphi = \cos(\varphi);\ c_\theta = \cos(\theta);\ c_\psi = \cos(\psi)$$
$$s_\varphi = \sin(\varphi);\ s_\theta = \sin(\theta);\ s_\psi = \sin(\psi)$$

It is important to highlight that there are many possible rotation sequences [3]. The one used in this thesis work is the (1,2,3) and, so, the above representation is true only in this case.

Quaternions

In $\mathbb{R}^3$, the rotation group $\mathcal{SO}(3)$ denotes the group of rotations around the origin under the operation of composition. Rotations are linear operations preserving vector length and the relative vector orientation. Generally, the rotation operation is performed using rotation matrices. The rotation matrices are not the only way to represent a rotation. However, an alternative way to represent the rotation group is introduced, i.e. the quaternions.
Briefly, for what concern this thesis work, a quaternion q is formed by a scalar term and vectorial one. So, it si a 4 elements vector:

$$\mathbf{q} = \begin{bmatrix} q_s \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} q_s \\ q_x \\ q_y \\ q_z \end{bmatrix} \tag{3.2}$$

And a rotation, defined by an axis $\mathbf{u}$ and an angle Θ, is represented as

$$\mathbf{q} = \begin{bmatrix} q_s \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} cos(\Theta/2) \\ \mathbf{u} sin(\Theta/2) \end{bmatrix} \tag{3.3}$$

It is important to highlight that the Hamilton notation is used here. Indeed, there are different definitions and conventions for the quaternions,

which depend on the order of the components, the multiplication definition (right or left handed) and the rotation of frames. A much more accurate explanation is given in the article of Joan Sola [27].
Moreover, a quaternion, in order to be part of the quaternion space $\mathcal{H}$ has to respect the following condition:

$$\mathbf{q}^T\mathbf{q} = q_s^2 + |\mathbf{q}_v|^2 = 1 \tag{3.4}$$

This is in someway the equivalent of the orthogonality constraint for the rotation matrices, for whic $R^TR = I = RR^T$. Of course there is also an expression linking the quaternions with the rotation matrices and viceversa [3]:

$$R(\mathbf{q}) = \begin{bmatrix} q_s^2 + q_x^2 - q_y^2 - q_z^2 & 2q_xq_y + 2q_sq_z & 2q_xq_z - 2q_sq_y \\ 2q_xq_y - 2q_sq_z & q_s^2 - q_x^2 + q_y^2 - q_z^2 & 2q_yq_z + 2q_sq_x \\ 2q_xq_z + 2q_sq_y & 2q_yq_z - 2q_sq_x & q_s^2 - q_x^2 - q_y^2 + q_z^2 \end{bmatrix} \tag{3.5}$$

$$\begin{aligned} q_s &= \frac{1}{2}\sqrt{1 + r_{11} - r_{22} - r_{33}} \\ q_x &= \frac{1}{4q_s}(r_{12} + r_{21}) \\ q_y &= \frac{1}{4q_s}(r_{31} + r_{13}) \\ q_z &= \frac{1}{4q_s}(r_{23} - r_{32}) \end{aligned} \tag{3.6}$$

Then, the rotation matrix R operating a rotation on the vector $\mathbf{v}$ is equivalent to the rotation using the quaternion $\mathbf{q}$

$$R\mathbf{v} = \mathbf{q} \odot \mathbf{v} \odot \mathbf{q}^* \,. \tag{3.7}$$

where $\odot$ indicates a quaternion multiplication and $\mathbf{q}^*$ is the quaternion conjugate operation. It is important to say that, generally, the quaternion multiplication is represented as $\otimes$, but in this work $\otimes$ will be used to represent the Kronecker product.

So now, the advantages of the quaternions with respect to the rotation matrices can be summarized:

- ✓ *More compact representation. Only 4 elements, instead of 9. At the same time, since there are 3 unknowns, it means to have only one constraint instead*

of six, as in the case of rotation matrices;

✓ *More numerically stable. Unit normalization of the quaternions presents less problems due to the rounding than the rotation matrix*;

✓ *Easier interpolation between quaternions*;

✓ *Faster multiplications*;

Disadvantages:

× *Less intuitive and more mathematical representation.*

By the way, for a comprehensive explanation on quaternion algebra, the author suggest consulting the work of Solà in [27].

3.2 Davenport's algorithm

The statement of the attitude determination is simple. However, there is a problem. In fact, the observation vectors (i.e. the measurements) are affected by some errors. This means that, generally, is not possible to find the exact rotation that brings them into the reference frame. Therefore, what can be done is to try to minimize a loss function defined by Wahba as:

$$J(R) = \frac{1}{2}\sum_{i=1}^{N} a_i \left|\mathbf{b}_i - R\mathbf{l}_i\right|^2 \tag{3.8}$$

Where $\mathbf{b}_i$ are the baselines in the body frame and $\mathbf{l}_i$ the baselines in the ECEF frame. While a_i are some positive weights. For simplicity they can be set in order to have

$$\sum_{i=1}^{N} a_i = 1$$

In fact, their sum will not affect in any way the minimization problem. They are generally chosen to be proportional to the baseline length. Thus, longer baselines are assumed to be more reliable.
To sum up, the problem in equation 3.8 is equal to find the optimal rotation that allow to have the best possible correspondence between the two set of vectors in a least-squares sense.

The optimal solution to the Wahba problem was proposed by Davenport, with the so called *q-algorithm* or *Davenport's algorithm*.

First of all, the minimization problem of equation 3.8 is translated into a maximization problem of the corresponding gain function

$$G(R) = 1 - J(R) = \sum_{i=1}^{N} a_i \mathbf{b}_i^T R \mathbf{l}_i = Tr[RB^T] \tag{3.9}$$

where

$$B = \sum_{i=1}^{N} a_i \mathbf{b}_i \mathbf{l}_i^T$$

as showed in Shuster's paper [25].

Now, using the quaternions, because of the advantages previously explained, the expression of the gain function of equation 3.9 becomes, after some operations [25]:

$$G(\mathbf{q}) = \mathbf{q}^T K \mathbf{q} \tag{3.10}$$

where K is a 4×4 matrix defined as

$$K = \begin{bmatrix} S - \sigma I & Z \\ Z^T & \sigma \end{bmatrix} \tag{3.11}$$

where:

- S is a 3×3 matrix obtained as $B + B^T$
- σ is the trace of B, i.e a scalar value
- Z is 3×1 matrix defined as the sum of the vector products of the reference and observation vectors

Thus, the problem to be solved is the one in equation 3.10, but taking into account the constrain of equation 3.4, because the final solution has to be part of the quaternion space $\mathbf{q} \in \mathcal{H}$. Through the usage of the Lagrange multiplier method, an equivalent unconstrained problem is found and the following equation is derived [25]:

$$K\mathbf{q} = \lambda \mathbf{q} \tag{3.12}$$

It is clear that this is an eigenvalue problem. Thus, the optimal quaternion is one of the eigenvectors of the K matrix. In particular it is the eigenvector corresponding to largest eigenvalue of K. This result is optimal in a least-squares sense.

3.2.1 QUEST algorithm

The Davenport's algorithm was tested and also used in some space missions, giving very good results for the attitude estimation. By the way, there was one problem. In fact, it was really expensive from a computational point of view and, so, time consuming (it was developed in 1970s). This could not be acceptable for daily attitude mission operations, like for the Magsat mission [26]. This problem is related to the fact that the eigenvalue/eigenvector problem is directly solved. For this reason a more efficient algorithm was developed, i.e. the QUEST (QUaternion ESTimator). It is an approximation of the Davenport's one but still it gives very good results, but in a faster way. The difference is that QUEST algorithm approximates the largest eigenvalue and, from that, it finds the corresponding optimal eigenvector.
Combining the original problem with the equation 3.10 an approximation for the optimal λ is found

$$\lambda_{opt} = 1 - J(\mathbf{q}) \tag{3.13}$$

Nevertheless, the quadratic loss function $J(\mathbf{q})$ should be very small for the optimal quaternion. For this reason a good approximation of the maximum eigenvalue of the K matrix should be $\lambda_{opt} = 1$. From this point a Newton-Raphson method can be applied to the characteristic equation of the matrix K

$$\det(K - \lambda I_4) = 0 \tag{3.14}$$

At the end, the final solution is expressed in function of the Rodrigues vector $Y = \frac{q_v}{q_s} = [(\lambda + \sigma)I - S]^{-1}Z$ as [26]:

$$\mathbf{q} = \frac{1}{\sqrt{(1 + |Y|^2)}} \begin{bmatrix} 1 \\ Y \end{bmatrix} \tag{3.15}$$

Of course, other algorithms have been developed, through the years, for solving the attitude problem. A list of some of the most used one is here reported:

- SVD (single value decomposition) [17]
- ESOQ (estimators of the optimal quaternion) [21]
- FOAM (Fast Optimal Attitude Matrix) [16]
- TRIAD [1]

The most robust algorithm is the Davenport's one. The others are less robust and are used when the time consumption has to be taken into account [18] . However, all these algorithm are able to give very good results, depending on the applications.
In this work, the QUEST and Davenport's algorithm have been used.

This problem can also be related to the GNSS positioning. In fact, in a multi-antenna GNSS system, finding the attitude corresponds to find the rotation relating baseline vectors across the body frame and the ECEF (Earth Centered Earth fixed) frame (equation 3.16). It is important also to give a definition of baseline. It is simply a vector connecting two different receivers (see Figure 3.4).

$$\mathbf{l}_e = R\,\mathbf{l}_b = \mathbf{q} \odot \mathbf{l}_b \odot \mathbf{q}^* \tag{3.16}$$

where R is the rotation matrix, $\mathbf{q}$ the quaternion, $\mathbf{l}_e$ the baseline in the ECEF frame and $\mathbf{l}_b$ the one in the body frame.

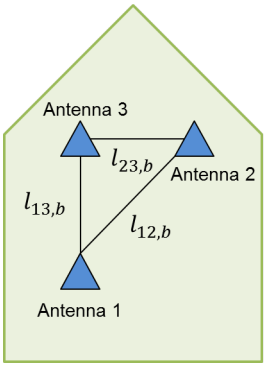

FIGURE 3.4: Example of a vessel with 3 antennas and, so, 3 baselines.

3.3 Factors affecting the attitude estimation

Now, let's see from a conceptual point of view what are the parameters that can affect the attitude estimation using the estimated GNSS positions of the receivers.
To find the optimal rotation the baselines in the ECEF frame are needed. This can be easily obtained by differencing the positions of the single receivers. Indeed, is possible to relate this problem to the general GNSS positioning, in case the individual position of each of the antennas is estimated using different positioning techniques, which give different performances:

- SPP (single point position) [24]
- PPP (precise point position) [11]
- RTK (Real-Time Kinematic) [4] [22]

Notice that they have been sorted starting from the less precise one.

Another factor affecting the accuracy of the attitude determination is the baseline length. This can be shown very easily through a geometrical approach [19]. In fact, suppose to have a situation in which there are only 2 antennas, i.e. only one baseline of length l, like the one in Figure 3.5.

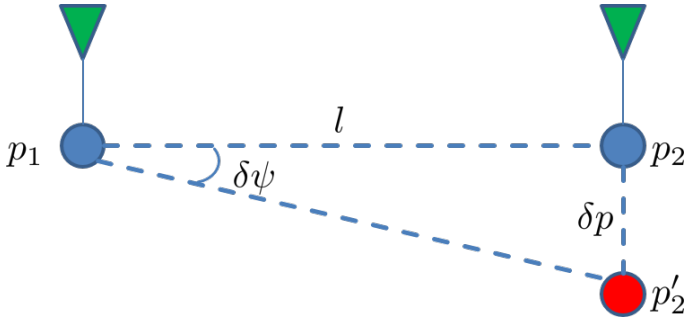

FIGURE 3.5: Representation of the geometrical approach scenario used for finding the estimation of the attitude error with respect to the baseline length l.

The heading error $\delta\psi$ can be simply estimated using the positioning error δp

$$\sin(\delta\psi) = \frac{\delta p}{\sqrt{l^2 + \delta p^2}} \tag{3.17}$$

Then, if l in much larger than the positioning error, the square root can be approximated to the baseline length. At the same time, the small angle assumption can be used, leading to the simple approximation :

$$\delta\psi \simeq \frac{\delta p}{l} \tag{3.18}$$

It is now easy to see that when l increases the error on the heading estimation will be smaller. In Figure 3.6 an example of this behaviour is showed.

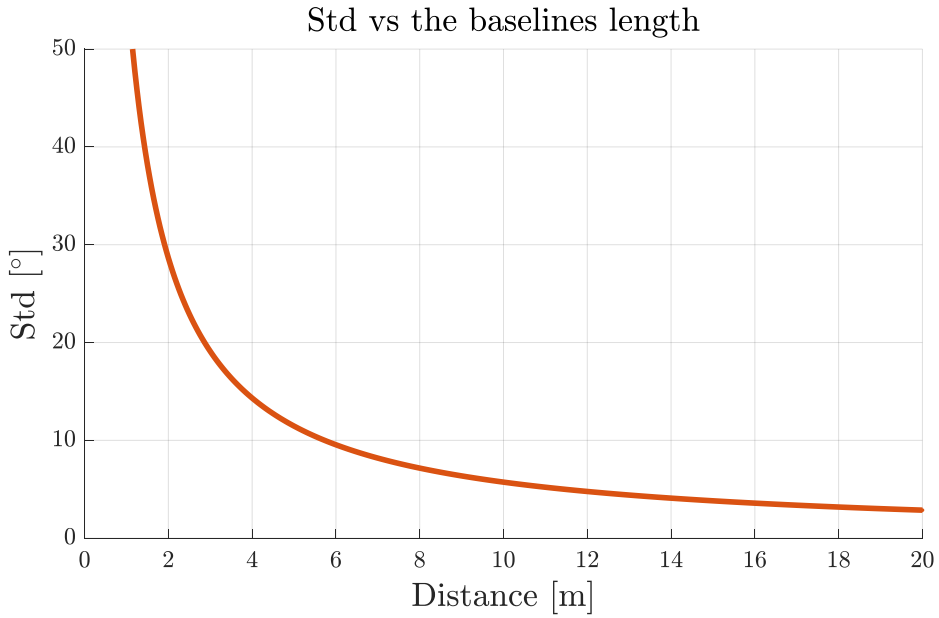

FIGURE 3.6: Example of the standard deviation of the heading error using the geometrical approach described above, changing the baseline length. An accuracy of 1 meter has been considered.

Another important factor regards the geometry of the antennas. As before, considering only 2 antennas, an intuitive proof that the geometry is very important can be given, looking at the car in Figure 3.7. In fact, it is easy to understand that a rotation around the y axis (pitch angle) can't be tracked. This is because the two antennas have the same x and z coordinates. So, in the xz plane, is like having a point. Thus, the pitch angle can't be estimated, because a point can't be rotated, but only shifted.

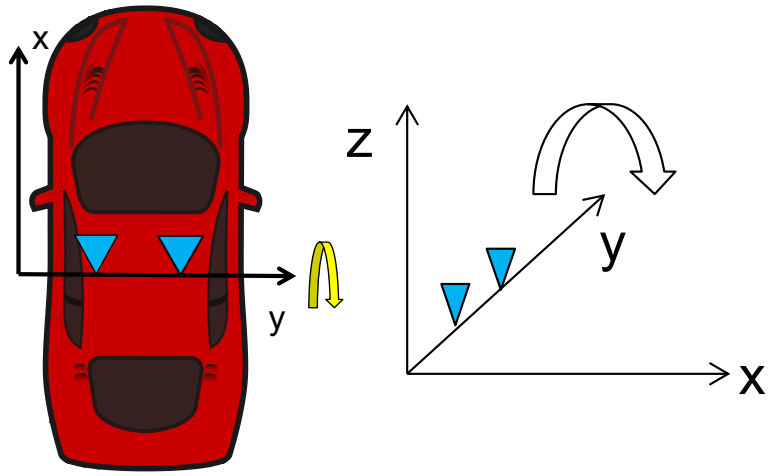

FIGURE 3.7: Example of a car with two antennas aligned over the y axis. In this case, the rotation around the y axis, i.e. the pitch angle can not be estimated, because there is not a vector to rotate.

This means that the geometry has to be taken into account to allow good results. By the way, the geometry will also depend on the applications, because of the shape of the vehicles and the importance of the angles to estimate. In fact, maybe, for cars the most important angle is the heading, while roll and pitch are less relevant. Therefore, the geometry could be chosen with this goal. However, for other applications, like in aviation, all the angles have to be well estimated, especially during the landing, to avoid a crash.

To summarize, the following factors could affect in some way the estimation of the attitude:

- Accuracy of the positioning;
- Baseline length;
- Geometry and number of the antennas.

The impact of all these factors will be analysed, through some simulations, in the result section (chapter 4).

Chapter 4

GNSS attitude problem using phase measurements

GNSS represents an appealing alternative for attitude determination, providing a drift-less absolute orientation solution while posing minimal requirements in terms of cost, weight and power consumption. Employing a setup of multiple GNSS antennas rigidly mounted on a vehicle, one seeks to find the rotation which relates the local and global navigation frames. Accurate attitude estimation based on GNSS requires the use of GNSS carrier phase observations.
Although GNSS attitude determination provides substantially higher accuracy than other systems, generally based on magnetic effects, its implementation poses some constraints. On one hand, at least a couple of GNSS receivers providing carrier phase are needed. Although receivers capable of providing carrier phase are currently costly, the technology for the production of low-cost receivers is rapidly growing. On the other hand, attitude accuracy is inversely proportional to the separation between the antennas, making this system impractical for small/miniaturized vehicles.

Previously, the attitude solving has been investigated, as well as the code and phase measurements and the working principle of GNSS. Now, the real GNSS attitude determination scenario has to be analyzed. A way to find a solution could be to estimate the position of the antennas and, then, compute the baselines (this is what is done in the first simulations of Chapter 5). However, for GNSS attitude determination, we are only interested in the relative orientation of the baselines and not really in the position of the antennas in the ECEF frame. Thus, the question is how to use as better as possible the GNSS measurements to get this information. As seen in Chapter 2, the code and phase measurements have many errors contribution. By the way, there is a mode to mitigate them, the so called *Double-Differencing (DD)*.

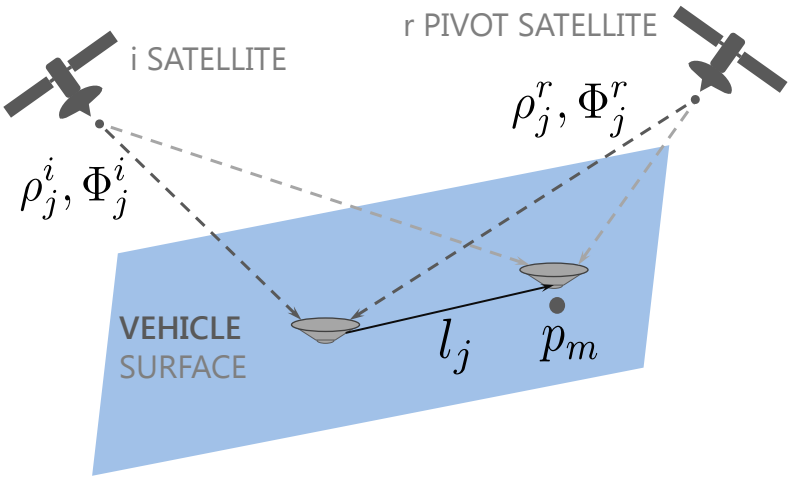

FIGURE 4.1: Scenario example for Double Differences computation.

Supposing to have $m+1$ antennas and $n+1$ GNSS tracked satellites in view (see example in Figure 4.1), first the Single Difference (SD) is computed. It means that, for each satellite in view, the difference between the measurements at two antennas are calculated. However, it is also very important to notice that the following results are good only under a short baseline assumption (shorter than 1 km) [5]. In fact, only in this case the signals can be said to have approximately the same path, which means to have very correlated atmospheric delays.
So, the SD, for the code measurements is:

$$\begin{aligned} \rho_j^i = \rho_j^i - \rho_m^i = \left\| p^i - p_j \right\| + c\delta t_j - c\delta t^i + Tr^i - I_0^i + \\ + \varepsilon_j^i - \left\| p^i - p_m \right\| - c\delta t_m + c\delta t^i - Tr^i + I_0^i - \varepsilon_m^i = \\ = \left\| p^i - p_j \right\| - \left\| p^i - p_m \right\| + c(\delta t_j - \delta t_m) + (\varepsilon_j^i - \varepsilon_m^i) \end{aligned} \tag{4.1}$$

And for the phase is:

$$\begin{aligned} \Phi_j^i = \Phi_j^i - \Phi_m^i = \left\| p^i - p_j \right\| + c\delta t_j - c\delta t^i + Tr^i - I_0^i + \lambda N_j^i + \\ + \varepsilon_j^i - \left\| p^i - p_m \right\| - c\delta t_m + c\delta t^i - Tr^i + I_0^i + \lambda N_m^i - \varepsilon_m^i = \\ = \left\| p^i - p_j \right\| - \left\| p^i - p_m \right\| + c(\delta t_j - \delta t_m) + \lambda(N_j^i - N_m^i) + (\varepsilon_j^i - \varepsilon_m^i) \end{aligned} \tag{4.2}$$

Where i indicates the i-th satellite, while j and m refers to the two antennas. In particular m is considered as the master antenna. It is also good to

notice that, from a model point of view , the only difference between code and phase is the presence of the ambiguities.
Thanks to the SD many of the error sources can be eliminated or enormously reduced. Then, also the receiver clock offset can be eliminated by choosing a pivot (or reference) satellite (usually the one with highest elevation) and computing the DD:

$$\begin{aligned} DD\rho_j^i = \rho_{jm}^r - \rho_{jm}^i = \left\|p^r - p_j\right\| - \left\|p^r - p_m\right\| + c(\delta t_j - \delta t_m) + \varepsilon_{jm}^r + \\ - \left\|p^i - p_j\right\| - \left\|p^i - p_m\right\| + c(\delta t_j - \delta t_m) - \varepsilon_{jm}^i = \\ = \left\|p^r - p_j\right\| - \left\|p^r - p_m\right\| - \left\|p^i - p_j\right\| + \left\|p^i - p_m\right\| + \varepsilon_j^i \end{aligned} \tag{4.3}$$

$$\begin{aligned} DD\Phi_j^i = \Phi_{jm}^r - \Phi_{jm}^i = \left\|p^r - p_j\right\| - \left\|p^r - p_m\right\| + c(\delta t_j - \delta t_m) + \lambda N_{jm}^r + \\ + \varepsilon_{jm1}^r - \left\|p^i - p_j\right\| - \left\|p^i - p_m\right\| + c(\delta t_j - \delta t_m) - \lambda N_{jm}^i - \varepsilon_{jm}^i = \\ = \left\|p^r - p_j\right\| - \left\|p^r - p_m\right\| - \left\|p^i - p_j\right\| + \left\|p^i - p_m\right\| + \lambda a_j^i + \varepsilon_j^i \end{aligned} \tag{4.4}$$

Where the notation ε_j^i is equivalent to $\varepsilon_j^i - \varepsilon_m^i$. The notation a_j^i stands for the double difference ambiguities.

This model can be linearized, considering the line of sight vectors [19]. Besides, considering the close distance between the receivers, one might consider the unit LOS vectors to be the same among the vehicle.

$$\left\|p^r - p_j\right\| = -u_r^\top \; p_j \tag{4.5}$$

So that the model for the double differences is :

$$DD\rho_j^i = -(u^i - u^r)^\top (p_j - p_m) + \varepsilon_j^i \tag{4.6}$$

$$DD\Phi_j^i = -(u^i - u^r)^\top (p_j - p_m) + \lambda a_j^i + \varepsilon_j^i \tag{4.7}$$

where a_j^i are the double difference ambiguities. However, the difference between the positions of the antennas represents, exactly, a baseline in the

ECEF frame!

$$l_j^e = (p_j - p_m) \tag{4.8}$$

Thus, this is the final expression for the DD model:

$$DD\rho_j^i = -(u^i - u^r)^\top l_j^e + \varepsilon_j^i \tag{4.9}$$

$$DD\Phi_j^i = -(u^i - u^r)^\top l_j^e + \lambda a_j^i + \varepsilon_j^i \tag{4.10}$$

It can be written in a more compact and general form [6]:

$$y = Aa + Bb + \varepsilon \tag{4.11}$$

where $a \in \mathbb{Z}^{m \cdot n}$ and $b \in \mathbb{R}^{3 \cdot m}$ are the unknowns, respectively the ambiguities and the baselines coordinates. $A \in \mathbb{R}^{2nm \cdot nm}$ and $B \in \mathbb{R}^{2nm \cdot 3m}$ are the design matrices, dependent respectively by λ and the unit LOS vectors u_i. ε is the observation noise. To conclude, y is the observation vector, formed by all the phase and code DD.

$$y = \begin{bmatrix} DD\Phi_1^{1:n} \\ \vdots \\ DD\Phi_m^{1:n} \\ DD\rho_1^{1:n} \\ \vdots \\ DD\rho_m^{1:n} \end{bmatrix} \tag{4.12}$$

where the notation $DD\Phi_m^{1:n}$ indicates all the phase measurements for the m-th baseline.

Given the observation model, a solution to this problem could be found applying a Least Squares (LS) (equation 4.13) [29]. Indeed, since there is a Gaussian noise, it should give an optimal solution.

$$\min_{a,b} \ \|y - (Aa + Bb)\|^2_{Q_{yy}} \tag{4.13}$$

where, in general

$$\|\cdot\|_Q = (\cdot)^T Q^{-1} (\cdot)$$

What can be noticed is that, in practice, a constraint over the ambiguities (which are known to be integer values) is applied.

However, solving this minimization problem, is not so trivial due to the integer nature of the ambiguities. Although no analytical solution exists for 4.13, in [33] it was suggested the decomposition of the LS into a sum of three consecutive least squares as follows:

$$\|y-(Aa+Bb)\|^2_{Q_{yy}} = \|e_r\|^2_{Q_{yy}} + \|\hat{a}-a\|^2_{Q_{\hat{a}\hat{a}}} + \left\|\hat{b}(a)-b(a)\right\|^2_{Q_{\hat{b}(a)\hat{b}(a)}} \tag{4.14}$$

where e_r represent the residuals. Q_{yy}, $Q_{\hat{a}\hat{a}}$ and $Q_{\hat{b}(a)\hat{b}(a)}$ are the variance covariance matrices. $\hat{a}$ and $\hat{b}(a)$ are the float solutions, i.e. the solution of the unconstrained problem.

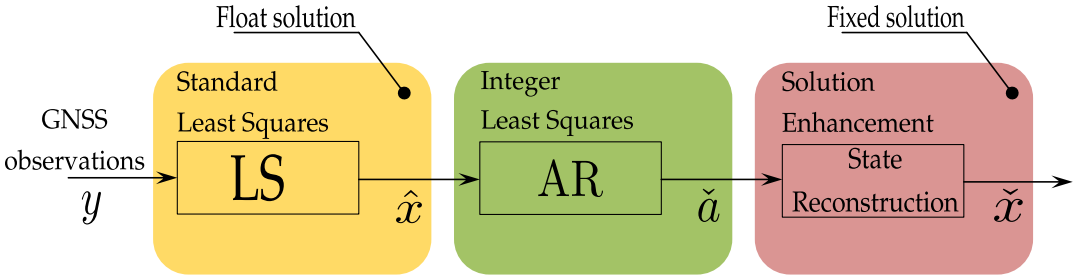

FIGURE 4.2: LAMBDA scheme.

So, to sum up the classical approach for finding the solution is the so called LAMBDA (Least squares AMBiguity Decorrelation Adjustment) method [30]. This procedure is illustrated on Figure 4.2.

The first stage is the computation of a standard least-squares adjustment, where the integer constrained is dropped, and the output estimation is often referred to as float solution. Then, the estimated float ambiguities and the corresponding variance-covariance matrix is used for the AR process. Among the methods for AR, one counts (sorted based on the quality of the results) [36] [28]:

- *Integer rounding*. Simply, all the float solutions are rounded to the nearest integer value.
- *Integer bootstrapping*. A sequential least squares adjustment is used in this case. The most precise ambiguity is rounded to the closest integer value. The other float solutions are then corrected by taking into account their correlation with the rounded one. And so on, until all the ambiguities are fixed.

- *Integer Least Squares (ILS)*. This is the optimal search method [32]. For this reason, its the methodology applied in this work. It consists of finding, in a search space defined through the variance-covariance matrix, an integer vector a which minimizes the distance from the float solution $\hat{a}$. See Figure 4.3 for a graphical representation of the working principle. The ILS will be explained in much more detail in the next section.

Finally, the computed integer ambiguities are used to improve the solution for the vector of dynamical parameters b. Such estimate is realized, once again in a least-squares sense, to obtain the fixed solution. The fixed solution generally inherits a much higher precision than the previously obtained float solution. The state reconstruction, or solution fixing, is finally accomplished.

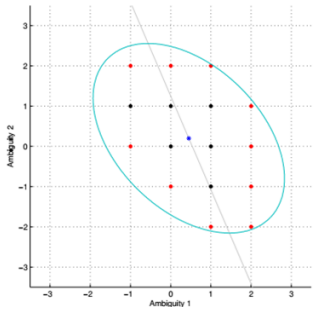

FIGURE 4.3: ILS search representation. [5]

Once the fixed solutions are computed the attitude can be easily determined. In fact, knowing the baselines in the ECEF frame b, the QUEST algorithm can be applied to find the rotation that carries them into the baselines in the body frame, as described in Chapter 3.

4.1 LAMBDA guide

As explained before, it is really important to have good fixed solutions for the baselines $\check{b}$. But, they depend on the fixed solution $\check{a}$. For this reason is really important to understand well, in detail, the working principle of the LAMBDA method. Since it could be tricky and not really easy to understand, a "LAMBDA guide" has been written, with the goal of providing a more easy and practical explanation with respect to the papers that can be found in the literature. To do this, a simple numerical example will be used, too. Since it is a simulation, the true values of the ambiguities (a) are

known.
Let's suppose to have 3 observations and, so, 3 ambiguities:

$$a = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}$$

Through a Least Squares (LS) the float (because they are real numbers) solutions are found:

$$\hat{a} = \begin{bmatrix} 5.45 \\ 3.10 \\ 2.97 \end{bmatrix} \quad Q_{\hat{a}\hat{a}} = \begin{bmatrix} 6.290 & 5.978 & 0.544 \\ 5.978 & 6.292 & 2.340 \\ 0.544 & 2.340 & 6.2880 \end{bmatrix}$$

where $Q_{\hat{a}\hat{a}}$ is the variance matrix.

However, the ambiguities are integer numbers. Using this information a more accurate estimation of the ambiguities can be obtained (fixed solution $\check{a}$). The LAMBDA method has the goal of searching for these integer vector. In practice the goal is to solve this Integer Least Square (ILS) problem [5]:

$$\check{a} = \min_{a \in \mathbb{Z}^n} \|\hat{a} - a\|^2_{Q_{\hat{a}\hat{a}}}$$

How is it done?

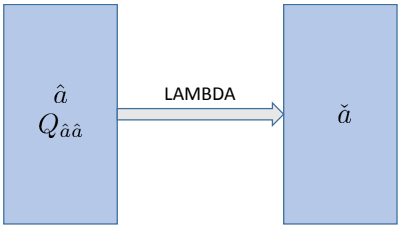

FIGURE 4.4: Input and final output of the LAMBDA method.

4.1.1 Decorrelation

Preliminary operations

The integer vector minimizer for the problem has to be found in the search space, defined as:

$$\Omega = \{\|\hat{a} - a\|^2_{Q_{\hat{a}\hat{a}}} \leq \chi^2\}$$

It is clear that this equation represents an n-dimensional hyper-ellipsoidal space, centered at $\hat{a}$, and whose shape depends on the variance matrix [5].

First of all, the ambiguities are highly correlated. For this reason, in order to have a more efficient search, a ***decorrelation*** is needed. Moreover, for convenience from a computational point of view, the integer part of the float solution is removed, so that all the values are between -1 and 1, since only the decimal part remains.

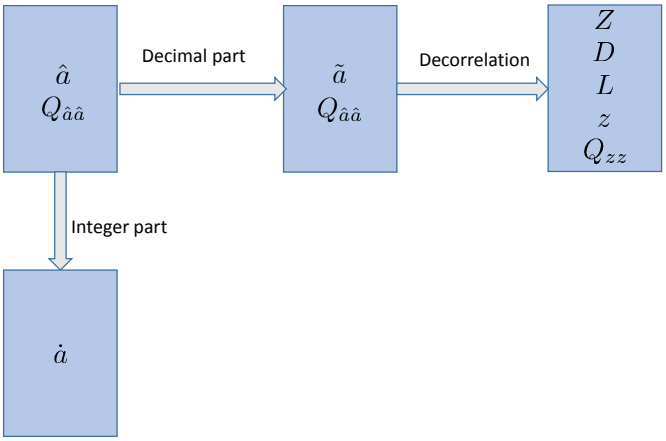

FIGURE 4.5: Processing of the float ambiguity and decorrelation

In our numerical example this means to have:

$$\hat{a} = \begin{bmatrix} 5.45 \\ 3.10 \\ 2.97 \end{bmatrix} \quad \tilde{a} = \begin{bmatrix} 0.45 \\ 0.10 \\ 0.97 \end{bmatrix} \quad \dot{a} = \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix}$$

So, the values used for the decorrelation are not the original float solutions $\hat{a}$, but their decimal part $\tilde{a}$ (see Figure 4.5).

Performing the decorrelation

To obtain the decorrelation a reparametrization is performed. Thus, the original ambiguities are transformed into new ones through a Z-transformation. For this reason the output of this operation, as can be seen in Figure 4.5, are [36]:

- Z, the matrix that defines the transformation;
- z, the transormed amiguities;
- Q_{zz}, the variance matrix of the new decorrelated ambiguities;

- L and D, the matrices used for the $L^T DL$-decomposition of $Q_{\hat{a}\hat{a}}$.

Supposing to have n ambiguities Z, Q_{zz}, L and D are n-by-n matrices. As a result, in the numerical example they simply are 3-by-3. In addition, it is important to highlight that D is a diagonal matrix and each value $d_j = D(j,j)$ represents the variance $\sigma^2_{\hat{a}_{j|j+1,..,n}}$ of the j-th ambiguity conditioned on all the next ones (i.e. from $j+1$ to n). Besides, the elements are sorted in order to have $d_n < ... < d_k < ... < d_1$. This is equal to say that the last ambiguity z_n is the most precise one. Then, L is the correlation matrix and it is a lower triangle one [36].
In the analyzed example these matrices are:

$$z = \begin{bmatrix} -1.57 \\ 2.02 \\ 0.35 \end{bmatrix} \quad L = \begin{bmatrix} 1 & 0 & 0 \\ 0.2677 & 1 & 0 \\ 0.3674 & 0.1310 & 1 \end{bmatrix}$$

$$D = \begin{bmatrix} 4.3102 & 0 & 0 \\ 0 & 1.1353 & 0 \\ 0 & 0 & 0.6260 \end{bmatrix} \quad Z = \begin{bmatrix} -2 & 3 & 1 \\ 3 & -3 & -1 \\ -1 & 1 & 0 \end{bmatrix}$$

where z, D and L are used in the search, whereas Z and Q_z are used only for the inverse transformation (see Figure 4.17 at page 51).

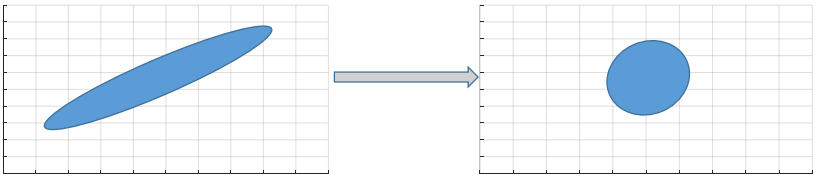

FIGURE 4.6: Graphical representation of the decorrelation of the search space in a 2D scenario.

4.1.2 ILS with shrinking search

Now let's see the most important part of this method, i.e. the search.

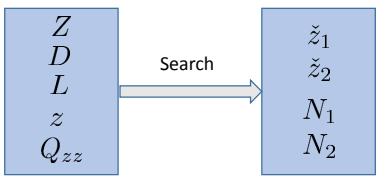

FIGURE 4.7: Inputs and outputs for the search operation.

As it is possible to see in Figure 4.7, this operation is carried out after the decorrelation. The matrices L, D and the ambiguities z are needed as input. The output are:

- the two best candidates $\check{z}_1$ and $\check{z}_2$;
- the square norms, N_1 and N_2, of the two candidates (the square norm is the metric used to evaluate the distance between the float solution and an integer vector of the search space).

The next pages will be completely focused on the explanation of the ***shrink and search algorithm***. As the name can suggest, the goal of this algorithm is to reduce the number of computations, by properly shrinking the search space.
In Figure 4.8 is showed a flow chart of all the operations performed. All the blocks and functions will be explained, in a general way, in different subsections.
Finally, the previously discussed numerical example will be analyzed, step by step, to give a more clear idea about the working principle of the algorithm.

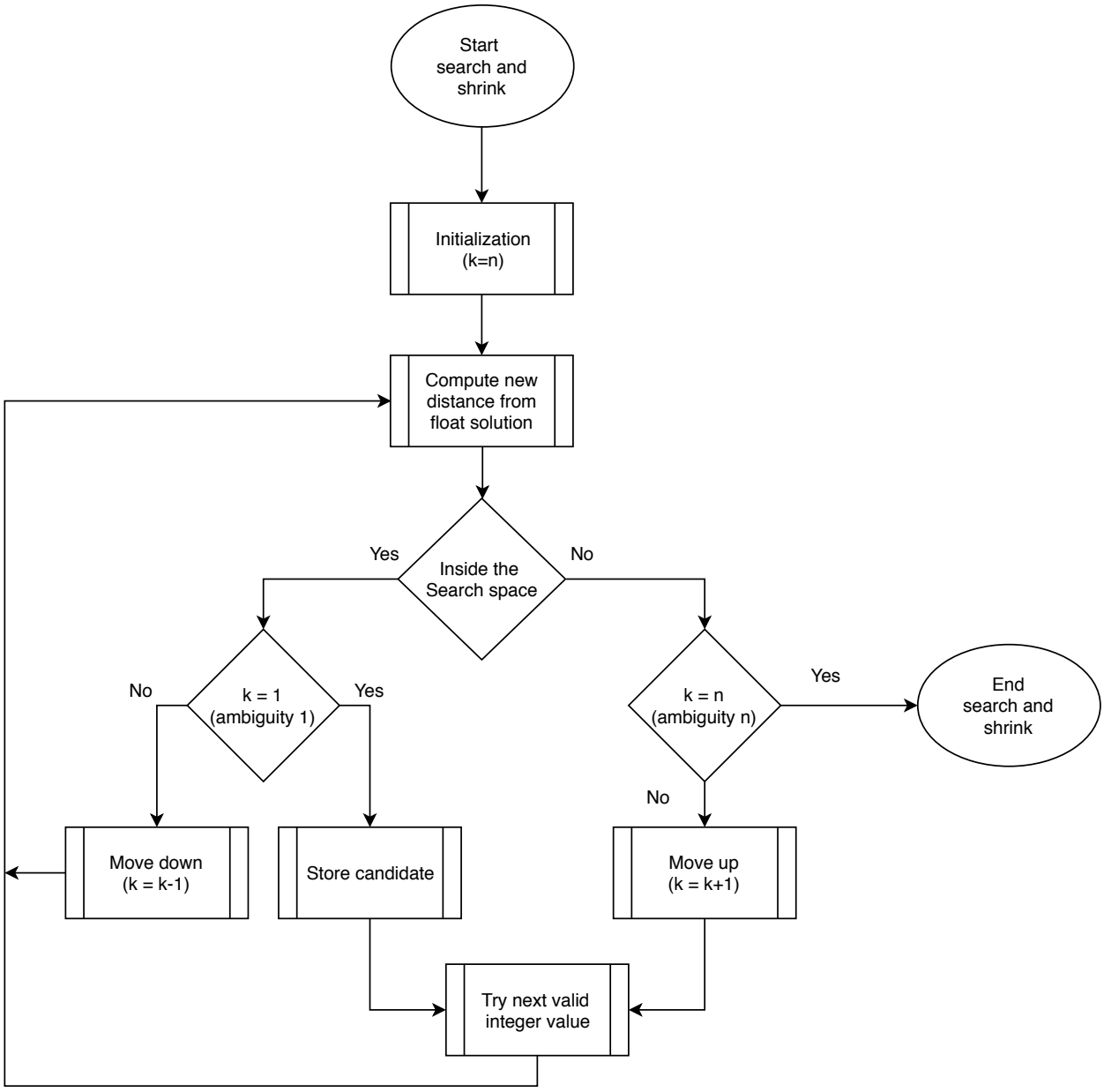

FIGURE 4.8: Flow chart of the "Search and shrink" algorithm. k is the counter used to take track of the current analyzed ambiguity.

Initialization

In Algorithm 1 are represented the steps of the initialization in the general case of n ambiguities.

All of them will be explained better and with the help of the numerical example.

Algorithm 1: Initialization for the search and shrink (general for n ambiguities).

1 $\chi^2 \longrightarrow \infty$;
2 $count = 0$;
3 $k = n$;
4 $S = O_{n,n}$;
5 $dist_n = 0$;
6 $z_n^C = z_n$;
7 $z_n^R = round(z_n)$;
8 $left = z_n^C - z_n^R$;
9 $step_n = sign(left)$;

Let's remember that, in the **example**, the transformed ambiguities are 3:

$$z = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} -1.57 \\ 2.02 \\ 0.35 \end{bmatrix}$$

All the parameters are initialized in the following way:

1. $\chi^2 = 10^{18}$
   Set the to a very high value (ideally infinite), like 10^{18}. In this way, the initial search space is large enough to contain at least two possible integer vector candidates for the ambiguities;

2. $count = 0$
   Counter to take track of the number of candidates found;

3. $k = 3$
   Counter used to track the current searched ambiguity;

4. $S = O_{3,3} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
   3-by-3 matrix used to compute the conditioned ambiguities. In the initialization step it is only a null matrix;

5. $dist = O_{3,1} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$
   This is the metric used to define the distance of the integer vector ambiguities from the corresponding float ones. The values used for the initialization will not affect the result, but for simplicity an all 0 vector is used;

6. $z_3^C = z_3 = 0.35$
   Due to the disposition of the ambiguities, the starting point will be the last ambiguity (the 3rd one z_3), because it is the most precise one (as written in section 4.1.1);

7. $z_3^R = round(z_3^C) = 0$
   There is the need to choose an integer number to start the search process for z_3. The choice is the one of rounding the conditioned ambiguity, which in this case is simply the original float solution. For this reason it is important to start with the most precise ambiguity!

8. $left = z_3^C - z_3^R = 0.35 - 0 = 0.35$
   In general $left$ represents the distance between a single conditioned ambiguity and the corresponding rounded one. In this case, it is clear that it refers to the third ambiguity z_3.
   However, notice that this difference is not weighted on the variance! This comment will be more clear in the next pages, as well as the usage of this variable;

9. $step_3 = sign(left) = 1$
   Generally, $step$ is a vector n-by-1 (3-by-1 in this example). It is fundamental for the success of the searching algorithm. The reason of that will be exhaustively explained in the next pages.
   Anyway, it is important to highlight that in case the sign of $left$ is 0 (it can happen only in the case $left = 0$), this value should be set to 1. If not, the search could not be able to find the best two candidates.

Computing the distance and verifying the belonging to the search space

After the initialization, the search loop starts.
The search is mainly based on two parameters, i.e. *newdist* and χ^2, which are compared, one to the other, at the beginning of the loop.

The metric *newdist* is computed in the following way:

$$newdist = dist_k + \frac{left^2}{d_j} \tag{4.15}$$

$$dist_k = \sum_{j=k+1}^{n} \frac{left^2}{d_j} \tag{4.16}$$

Where $dist_k$ is the distance of the last integer ambiguities (from $k+1$ to n) with respect to the conditioned float solutions (check the definition of the $left$ variable).
This means that *newdist* is the distance referred to the ambiguities from k to n. It is really important to notice the presence of the element d_j. In fact, it is the j-th element of the diagonal matrix D ($d_j = D(j,j)$) and, as explained before, it represents the variance $\sigma^2_{j|j+1,\dots,n}$ of the j-th ambiguity conditioned on all the successive ones.
Once this is done, this metric is compared to the χ^2. Of course there are two possibilities:

- $newdist < \chi^2$
  This means that we are inside the ellipsoidal search space. Thus, a good candidate could still be found. This condition will lead either to the "move down" function or to the "store candidate" one;

- $newdist \geq \chi^2$
  This means that we are out of the ellipsoidal search space and, so, a candidate can't be found for this combination of integer ambiguities! This condition, dependently from the actual value of k, will lead either to the "move up" function or to the "end search" one.

A pseudo code of the searching loop is represented (algorithm 2), to better understand the conditions and the order of execution of all the functions.

It will possible to better understand it in the practical example with the numerical values.

Algorithm 2: While loop for the general case of the search and shrink algorithm.

```
Initialization;
while search do
    COMPUTE newdist;
    if newdist < χ² then
        if k == 1 then
            Function STORE CANDIDATE;
            Function TRY NEXT VALID INTEGER;
        else
            Function MOVE DOWN (k ⟵ k − 1);
        end
    else
        if k == n then
            Function END SEARCH/EXIT FROM WHILE LOOP;
        else
            Function MOVE UP (k ⟵ k + 1);
            Function TRY NEXT VALID INTEGER;
        end
    end
end
```

Before diving in the next steps, it is better to highlight something that could be tricky. Indeed, during the search, the ambiguities will be seen in a different order from the usual vectorial form $\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$. In fact, since the starting point is the last ambiguity, the disposition is like in Figure 4.9. ***The expressions "move down" and "move up" refer to this displacement***. Surely, this is something used only for convenience and it is not mandatory.

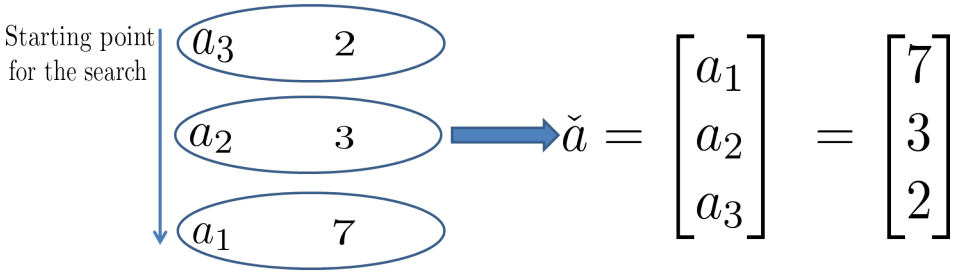

FIGURE 4.9: Graphical representation of the ambiguities during the search algorithm. Notice that it is not the same as their positions in the classical vectorial form $\check{a}$.

Move down

When inside the search space and when the distance is not computed over all the ambiguties (i.e. $k \neq 1$), the "move down" operations are performed. As the name suggests the counter k is decremented and this means that the next element $(k-1)$ of the ambiguity vector will be taken into account (see Figure 4.10). The current distance is stored and an update of some variables is realized.

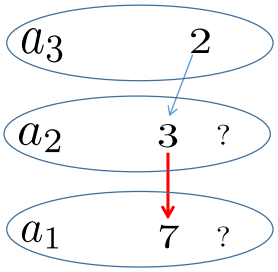

FIGURE 4.10: "Move down" example. For the integer values $[a_3; a_2] = [2; 3]$ we are still inside the search space. We have to "move down" to check if, for $a_1 = 7$, we can have an integer vector candidate.

Here the S matrix is finally used to compute the conditioned ambiguity z_k^C. Let's remember that S is a $n \times n$ matrix (3×3 in the example). A graphical representation of the S matrix is given in Figure 4.11. It is easy to understand why it is like this. In fact, we use the last row (n) to compute, employing the correlation matrix L and the variable $left$, the previous row $(n-1)$. Then, the row $n-1$ is used to get the row $n-2$ and so on, until the first row. Therefore, a row will be dependent on all the previous ones, like a cascade. For example, the j-th row will depend on all the rows from $j-1$ to n. In the meanwhile, the conditioned ambiguity is computed as

$z_k^C = z_k + S(k,k)$. In this way, the relationship of dependency among all the ambiguities, previously described, is always preserved.

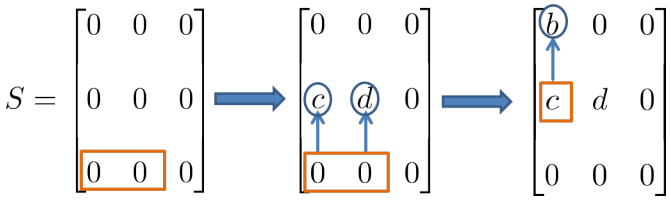

FIGURE 4.11: S matrix update in a 3 ambiguities example. b, c and d are real numbers. Notice the cascade effect. The 1st row depends on the 2nd one, which in turn depends on the 3rd one.

After the update of S and the calculation of the conditioned ambiguity, the integer ambiguity z_k^R is got, through a rounding. New values for the variables $left$ and $step_k$ are found.

Store candidate

If we are inside the ellipsoid and the ambiguity for which the distance has been computed is the first one (z_1), it means that a complete integer vector has been found (see Figure 4.12). As a result, it is stored as suitable candidate and its distance from the float solution is saved. Moreover, if this is not the first candidate, an important operation has to be done. A new value for χ^2 is set, equal to the distance of the current second best candidate from the float solution. In this way, the search space is enormously reduced. If there is a candidate inside this new ellipsoid, then, it will be surely better than the current one (see Figure 4.13).

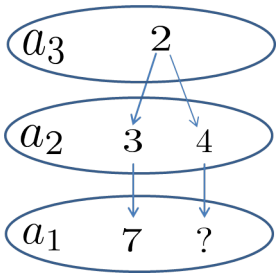

FIGURE 4.12: Example: store the candidate [7; 3; 2].

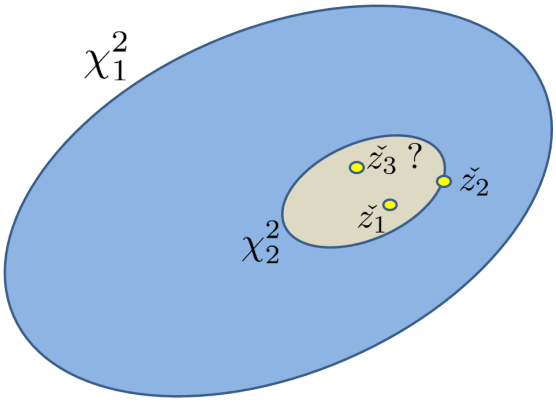

FIGURE 4.13: *Shrinking the search space.* $\check{z}_2$ is at the edge of the new ellipsoid, since it is defined by its distance χ_2^2. If another candidate, $\check{z}_3$, is found in this new ellipsoid, then, it will be surely better than $\check{z}_2$ in terms of distance from the float solution. Therefore, $\check{z}_2$ will be discarded and a new, smaller, search space will be defined. The procedure will stop when an ellipsoidal search space with only 2 candidates inside will be found.

Move up

This function is executed when there is a so called "dead end" for an ambiguity which is not the last one (z_n). It means that the vector is already out of the ellipsoid and, so, there is not the possibility of finding a proper candidate with those integer values (see Figure 4.14). For this reason, the counter k is incremented by 1 ($k = k + 1$) and a new integer value for z_{k+1} will be tested.

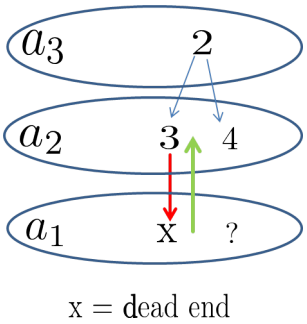

FIGURE 4.14: "Move up" example. A dead end is found, i.e. is not possible to find a candidate for that combination of integer values. The only solution is to "move up" to the previous ambiguity a_2 and try a new integer value for it.

Try next valid integer

It is executed either after "store candidate" or "move up". The only difference is that in the first case there is always $k = 1$, i.e. the first ambiguity z_1^R is modified (see Figure 4.15). On the contrary, in the second one, any other value of k is possible. Thus, this function is carried out every time there is the need to extend the search to other possible integer values for an ambiguity.

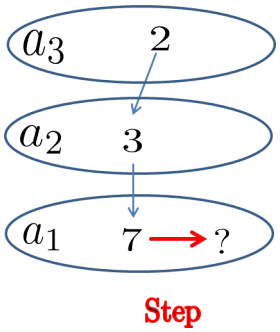

FIGURE 4.15: Try next valid integer example after "store candidate".

The procedure is really simple:

- Find next integer value z_k^R to test, using the $step_k$ variable, i.e. $z_k^R = z_k^R + step_k$ (see Figure 4.16);
- Update the $left$ variable;
- Update the $step$ variable using its previous value (see Figure 4.16 for a better understanding).

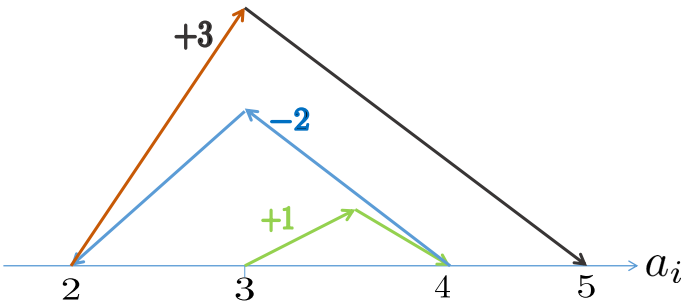

FIGURE 4.16: Example: choosing the next integer value to test for an ambiguity. Here it is also clear how the *step* variable is updated. At the beginning $a_i = 3$ and $step = 1$. Thus, the integer value to test becomes $a_i = 4$. At this point *step* will be updated to the value -2 and, so, at the next execution of the "try next integer" function for that particular ambiguity, the integer value to be tested will become 2. It is clear that the goal is to scan the set of the integer numbers in both the directions, starting from the initial value ($a_i = 3$ in this graphical example).

End search

This can happen only if we are out of the ellipsoid and with the condition of taking into account only the last ambiguity (z_n).
In easy words, this means that you are already too far from the float solution, while considering only the distance of the most precise ambiguity (z_n). Therefore, it is not possible to find any new suitable integer vector candidate. **The search has to end**.

Following the example

Now that all the operations and the motivations have been explained, the numerical example will be analyzed step by step to make the search algorithm more clear.
In subsection 4.1.2 the *initialization* for the example has already been presented. That one is the starting point. Consequently, in this subsection, only the search loop will be analyzed.
The suggestion is to check, in the meanwhile, the flow chart of Figure 4.8 at page 35 or the Algorithm 2 at page 39.

Inizialization

- See subsection 4.1.2;

Compute newdist

- newdist = 0.1957;

Is it inside the search space?

- $\chi^2 = 10^{18} \Longrightarrow newdist < \chi^2 \Longrightarrow$ Inside the search space

What is the value of k? Is $k = 1$?

- No, $k = 3 \Longrightarrow$ Execute "Move down"

Move down

- Decrement $k \Longrightarrow k = k - 1 \Longrightarrow k = 3 - 1 = 2$
- Save the distance $\Longrightarrow dist_2 = newdist = 0.1957$;
- Update S matrix (used for the conditioned ambiguities) $\Longrightarrow \begin{bmatrix} 0 & 0 & 0 \\ -0.1286 & -0.0458 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- Compute the conditioned ambiguity $\Rightarrow z_2^C = z_2 + S(2,2) = 2.02 + (-0.0458) = 1.9742$
- Round the conditioned ambiguity $\Rightarrow z_2^R = round(z_2^C) = round(1.9742) = 2$
- Compute $left \Rightarrow left = z_2^C - z_2^R = 1.9742 - 2 = -0.0258$
- Find $step_2 = sign(left) = -1$

Compute newdist

- newdist = 0.1963;

Is it inside the search space?

- $\chi^2 = 10^{18} \Longrightarrow newdist < \chi^2 \Longrightarrow$ Inside the search space

What is the value of k? Is $k = 1$?

- No, $k = 2 \Longrightarrow$ Execute "Move down"

Move down

- Decrement $k \Longrightarrow k = k - 1 \Longrightarrow k = 2 - 1 = 1$
- Save the distance $\Longrightarrow dist_1 = newdist = 0.1963$;
- Update S matrix (used for the conditioned ambiguities) $\Longrightarrow \begin{bmatrix} -0.1217 & 0 & 0 \\ -0.1286 & -0.0458 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- Calculate the conditioned ambiguity $\Rightarrow z_1^C = z_1 + S(1,1) = -1.57 + (-0.1217) = -1.6917$
- Round the conditioned ambiguity $\Rightarrow z_1^R = round(z_1^C) = round(-1.6917) = -2$
- Compute $left \Rightarrow left = z_1^C - z_1^R = -1.6917 - (-2) = 0.3083$
- Find $step_1 = sign(left) = 1$

Compute newdist

- newdist = 0.2183;

Is it inside the search space?

- $\chi^2 = 10^{18} \Longrightarrow newdist < \chi^2 \Longrightarrow$ Inside the search space

What is the value of k? Is $k = 1$?

- Yes, $k = 1 \Longrightarrow$ Execute "Store candidate"

Store candidate

- What is the value of *count*? $count = 0 \Rightarrow$ This is the first candidate we have found.
- Increment $count \Rightarrow count = count + 1 = 1$;
- Save the candidate and its square norm $\Rightarrow \check{z}_1 = z^C = \begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix}$ and $metric_1 = newdist = 0.2183$
- execute "try next integer value"

Try next integer value

- Try the next integer for $z_1^R = z_1^R + step_1 = -2 + 1 = -1$, because $k = 1$
- Compute $left = z_1^C - z_1^R = -0.6917$
- Update $step_1 = -step_1 - sign(step_1) = -2$

Compute newdist

- newdist = 0.3073;

Is it inside the search space?

- $\chi^2 = 10^{18} \Longrightarrow newdist < \chi^2 \Longrightarrow$ Inside the search space

What is the value of k? Is $k = 1$?

- Yes, $k = 1 \Longrightarrow$ Execute "Store candidate"

Store candidate

- What is the value of *count*? $count = 1 \Rightarrow$ This is the second candidate we have found

- Save the candidate and its square norm $\Rightarrow \check{z}_2 = z^C = \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix}$ and $metric_2 = newdist = 0.3073$
- Update $\chi^2 \Rightarrow \chi^2 = max(metric_1, metric_2) = max(0.2183, 0.3073) = 0.3073$
  We are sure there are at least 2 candidates in the search space defined by the new value of χ^2. If we are able to find another candidate inside this ellipsoid, then it will be the new (second or first) best candidate.
- execute "try next integer value"

Try next integer value

- Try the next integer for $z_1^R = z_1^R + step_1 = -1 - 2 = -3$, because $k = 1$
- Compute $left = z_1^C - z_1^R = 1.3083$
- Update $step_1 = -step_1 - sign(step_1) = 3$

Compute newdist

- newdist = 0.5934;

Is it inside the search space?

- $\chi^2 = 0.3073 \Longrightarrow newdist > \chi^2 \Longrightarrow$ Outside the search space!

What is the value of k? Is $k = 3$?

- No, $k = 1 \Longrightarrow$ Execute "Move up"

Move up

- Increment $k \Longrightarrow k = k + 1 = 2$
- execute "try next integer value"

Try next integer value

- Try the next integer for $z_2^R = z_2^R + step_2 = 2 - 1 = 1$, because k=2;

- Compute $left = z_2^C - z_2^R = 0.9742$
- Update $step_2 = -step_2 - sign(step_2) = 2$

Compute newdist

- newdist = 1.0316;

Is it inside the search space?

- $\chi^2 = 0.3073 \Longrightarrow newdist > \chi^2 \Longrightarrow$ Outside the search space!

What is the value of k? Is $k = 3$?

- No, $k = 2 \Longrightarrow$ Execute "Move up"

Move up

- Increment $k \Longrightarrow k = k + 1 = 3$
- execute "try next integer value"

Try next integer value

- Try the next integer for $z_3^R = z_3^R + step_3 = 0 + 1 = 1$, because k=3;
- Compute $left = z_3^C - z_3^R = -0.6500$
- Update $step_3 = -step_3 - sign(step_3) = -2$

Compute newdist

- newdist = 0.6749;

Is it inside the search space?

- $\chi^2 = 0.3073 \Longrightarrow newdist > \chi^2 \Longrightarrow$ Outside the search space!

What is the value of k? Is $k = 3$?

- Yes, $k = 3 \Longrightarrow$ Execute "End search"

End search

- Sort the two found candidates based on their square norms and return them $\Longrightarrow \check{z}_1 = \begin{bmatrix} -2 \\ 2 \\ 0 \end{bmatrix}$ and $\check{z}_2 = \begin{bmatrix} -1 \\ 2 \\ 0 \end{bmatrix}$.

Back-transformation

- Come back to the original domain. Change again the parameters, from z to a (see section 4.1.1) $\Rightarrow \check{a} = Z^{-1} \cdot \check{z} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & -1 \\ 0 & 1 & -3 \end{bmatrix} \cdot \begin{bmatrix} -2 & -1 \\ 2 & 2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 2 & 2 \end{bmatrix}$
- Let's remember, from section 4.1.1, that $\dot{a} = \begin{bmatrix} 5 \\ 3 \\ 2 \end{bmatrix}$
- Finally, the best candidate is: $\bar{a}_1 = \check{a}_1 + \dot{a} = \begin{bmatrix} 5 \\ 3 \\ 4 \end{bmatrix}$
- The second best candidate is: $\bar{a}_2 = \check{a}_2 + \dot{a} = \begin{bmatrix} 6 \\ 4 \\ 4 \end{bmatrix}$

In figure 4.17 the general scheme of the operations for the LAMBDA method is showed to make more clear the last part of the example.

4.1.3 Ratio test. Open question. Empirical approach

Ratio test

Using the previously explained method is it possible to find the two best candidates, but how to use them? How is it possible to know if the best candidate is good or not? In fact, the integer ambiguity fixing should also involve an acceptance test on the integer solution. This can be done through

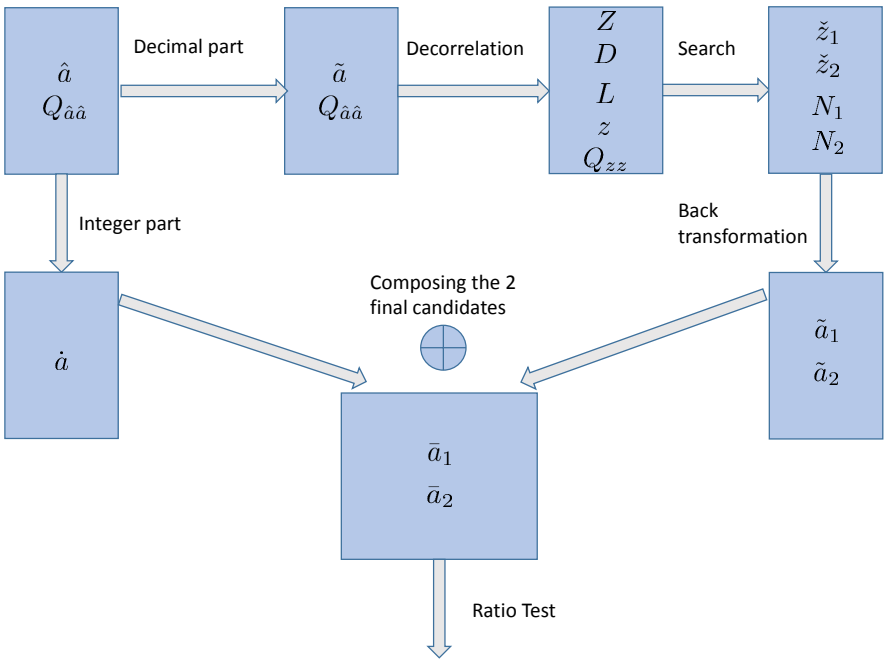

FIGURE 4.17: General scheme for LAMBDA method.

a Ratio Test, i.e. a discrimination test to check how much closer is the best integer candidate to the float solution, compared to the second best one. In practice, indicating with N_1 and N_2 the metric of the two candidates $\bar{a}_1$ and $\bar{a}_2$, what is done is to check if:

$$\frac{N_2}{N_1} \geq \gamma \tag{4.17}$$

where γ is the threshold value, which is often a fixed value like 2 or 3 [36]. This value determines the size of the acceptance regions (see Figure 4.18). Intuitively and with the help of *detection theory* analysis, it is easy to understand that a larger value of γ corresponds to a lower false alarm probability, but, at the same time, to a lower detection probability.

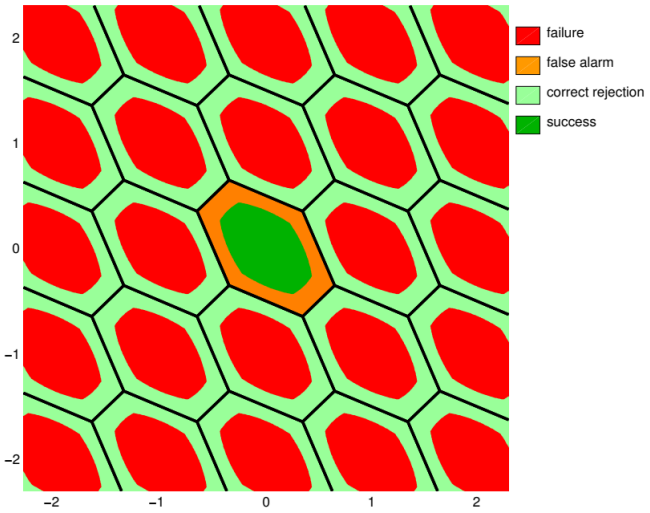

FIGURE 4.18: Acceptance regions with Ratio Test [36].

Open questions

However,some questions about the choice of the threshold should be asked.

- First of all, is it dependent on the number of the double differences and, so, of the ambiguities present in the ambiguity vector? In fact, an higher number of ambiguities should imply more integer values to test. Can this affect the probability of having a second best candidate which is more far from the first one, or not? Probably yes, but its impact on the performances should be properly analyzed.
- Does it depend on the *max* e *min* values of the float ambiguities? No, because, as showed before, the algorithm is applied for the decimal part of the float ambiguity only.

Setting the threshold

Finally, how to properly set a value for the threshold?
As previously said, there is not a correct way to do it. It all depends on the false alarm probability that we want to achieve. Therefore, this would depend on a statistical analysis. In fact, an higher threshold means lower false alarm, but also a lower detection! But, what could be done is to use an empirical approach. A large number of Montecarlo simulations for different ambiguity vectors, in order to have statistically meaningful results, and compare the ***success rate*** with the ***fixing rate***! The first one is the percentage of times in which the best candidate is equal to the true ambiguity

vector, while the latter, represents the times in which the ratio is bigger than the threshold γ and, consequently, the best candidate is accepted as a good solution. In this way, if the fixing rate is much lower than the success rate it means that the condition on the ratio test is too tight, hence the value of γ should be decreased. On the contrary, if the success rate is lower than the fixing rate the threshold is too low.

Finally, the results for different ambiguity vector sizes should be compared, in order to check how much this parameter is important.

4.2 A new approach. Quaternion-based Constrained LAMBDA

A new approach will now be proposed. In fact, from equation 4.9 and 4.10 the double difference is

$$DD\rho_j^i = -(u^i - u^r)^\top l_j^i + \varepsilon_j^i$$
$$DD\Phi_j^i = -(u^i - u^r)^\top l_j^i + \lambda a_j^i + \varepsilon_j^i$$

But the baseline in the ECEF frame can be seen as the baseline in body frame to which a rotation is applied. As said in the prevoius chapters, a rotation can be represented in different ways. One of them is using the quaternions q

$$l_j^e = R l_j^b = q \odot l_j^b \odot q^*$$

where $\odot$ is the quaternion multiplication. Usually, it is denoted as $\otimes$, but in this thesis that symbol would collide with the Kronecker product. Now, the new expression for the DD model is

$$DD\rho_j^i = -(u^i - u^r)^\top q \odot l_j^b \odot q^* + \varepsilon_j^i$$
$$DD\Phi_j^i = -(u^i - u^r)^\top q \odot l_j^b \odot q^* + \lambda a_j^i + \varepsilon_j^i$$

Tracking the quaternion orientation allows for exploiting the prior information on the baselines length, as well as the relative orientation between them. As explained, this is a new approach, which will be called "Quaternion approach" or "Tracking quaternion" in the rest of the thesis. In practice

the first advantage that can be seen is that it is already constrained by the baseline length! But let's see what are really the differences with respect to the classical approach. To do this, the models have to be defined first.

The model can still be written in a more compact form with respect to the one in the previous sections. In fact it can be written as:

$$y = Hx + \varepsilon \tag{4.18}$$

where H is the Jacobian matrix of the observation model, y the observations and x the unkowns. The observation is always the union of the code and phase measurements.

But, for H and x the expression is different based on the kind of approach used.

- *Classical approach.*

$$\min_{b \in \mathbb{R}^{3m \times 1},\, a \in \mathbb{Z}^{mn \times 1}} \|y - Hx\|^2_{Q_y} \tag{4.19}$$

The state x contains all the baselines (it depends on the number of antennas) and, of course, the ambiguity vector a

$$x = \begin{bmatrix} b_1 \\ \vdots \\ b_m \\ a \end{bmatrix} \tag{4.20}$$

The Jacobian is instead (for single frequency case)

$$H = \begin{bmatrix} H_\Phi \\ H_\rho \end{bmatrix} = \begin{bmatrix} I_m \otimes U & I_m \otimes \lambda I_n \\ I_m \otimes U & I_m \otimes O_n \end{bmatrix} \tag{4.21}$$

where the symbol $\otimes$ in this case is the Kronecker product. I_m and O_n represents respectively the m-size unitary matrix and the n-by-n null matrix. U is the m-by-3 matrix fo the LOS vectors of the double-differences:

$$U = \begin{bmatrix} -(u^1 - u^r)^\top \\ \vdots \\ -(u^m - u^r)^\top \end{bmatrix} \tag{4.22}$$

In Figure 4.19 a less compact, but easier to understand, representation of the H matrix is shown in the case of 3 baselines.

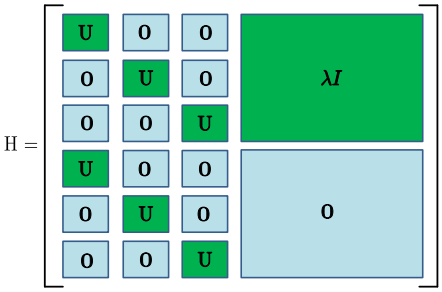

FIGURE 4.19: Representation of the H matrix for the classical approach in the case of 3 baselines, without the usage of the Kronecker product.

- *Quaternions approach.*
  To avoid redundancy notation problems the parameters for this approach will be defined with a $'$ sign. Thus

$$\min_{q \in \mathcal{H},\, a \in \mathbb{Z}^{mn \times 1}} \left\| y - H'x' \right\|^2_{Q_y} \tag{4.23}$$

The state x' contains the quaternion and, again, the ambiguity vector a:

$$x' = \begin{bmatrix} q \\ a \end{bmatrix} \tag{4.24}$$

The Jacobian is

$$H' = \begin{bmatrix} H'_\Phi \\ H'_\rho \end{bmatrix} = \begin{bmatrix} J_{m,1} \otimes U\, F_q^{1:m} & I_m \otimes \lambda I_n \\ J_{m,1} \otimes U\, F_q^{1:m} & I_m \otimes O_n \end{bmatrix} \tag{4.25}$$

where $J_{m,1}$ is a m-by-1 all ones vector. U is the same as in the classical approach, i.e. the matrix of the unit LOS vectors of the DD. Finally, F_q is a 3×4 matrix defined as the derivative of the rotated baseline in the body frame with respect to the quaternion itself:

$$F_q^i = \frac{\partial(q \odot b_i \odot q^*)}{\partial q} \tag{4.26}$$

For a comprehensive explanation on quaternion algebra, the author suggest consulting the work of Solà in [27].
In Figure 4.20 the H' matrix is written in a less compact to form, in the case of 3 baselines, to allow a better understanding.

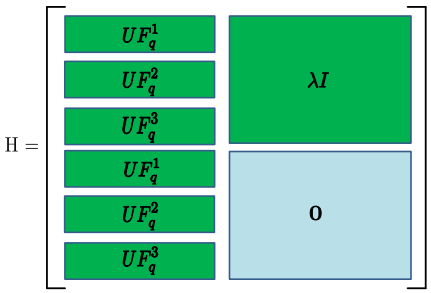

FIGURE 4.20: Representation of the H matrix for the quaternion approach in the case of 3 baselines, without the usage of the Kronecker product.

Now that the matrices have been explicitly derived, an analysis of the advantages and disadvantages of this new apporach can be easily explained.
Advantages:

- ✓ *Higher redundancy of observations*. This come from the difference in the state vector. While the number of ambiguities remains the same, the number of unknowns for the quaternion approach is independent from the number of the baselines. On the contrary, in the classical approach there is the presence of 3 unknowns for each baseline. This results in a larger number of unknowns, but with the same amount of DD equations.
- ✓ *Better float solution*. Since in the quaternion approach the baseline length and relative orientation is preserved.
- ✓ *Better for multi-sensory systems*. It can be easily integrated with other kinds of measurements, e.g. gyroscopes. [13] [14] [23]

× *Non-linear equation for the float finding procedure.* Unlike the estimation of the baselines in the ECEF frame, where the equations are fully linear, the quaternion equations results in a non-linear problem, which is to be either estimated iteratively using a Gaussian-Newton method, or in a single step using a Recursive estimator (i.e., an Extended Kalman Filter).

Finally, in the next chapter the results obtained through the simulations will be showed and analyzed.

Chapter 5

Simulations and results

In this last chapter, the results obtained will be showed and explained. For the simulations and the plots Matlab has been employed. In particular, the first two sections refers to the topics of the Chapter 3, whereas the last section regards the Chapter 4.

5.1 Attitude Determination using Vector Observations

A simulation has been set. There is a rigid body (a vessel in this case), with a body frame with axis x, y and z, which are respectively the roll, pitch and heading (or yaw) axis. The system is displayed in Figure 5.1.

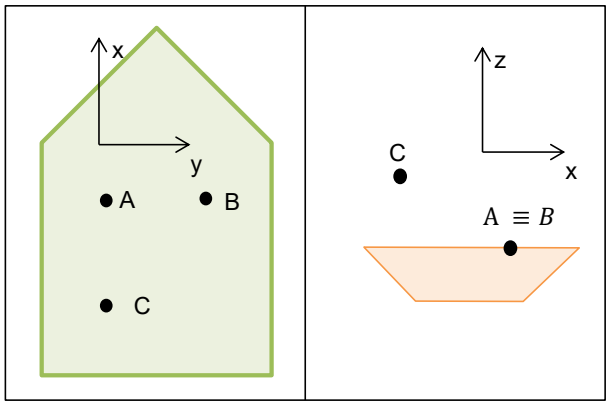

FIGURE 5.1: Body frame definition for testing the attitude solver. A "triangular geometry" for the antennas is used.

There are 3 antennas (A, B and C), which means to have 3 baselines (AB, AC, BC). A and B are alligned over the y axis and share the same height (z coordinate). On the contrary, C is aligned over the x axis with A and has a different height. Moreover, the baseline lengths are all the same ($5m$). This

kind of configuration has been chosen because it is simple and the antennas are geometrically well distributed. Indeed, as explained in Chapter 2, the geometry used should affect in someway the results.
Knowing the position in the body frame is, then, possible to simulate the position in the ECEF frame by: adding some random noise, computing the new baselines and applying a rotation to them.

First, a random zero mean Gaussian noise is added to each of the antenna ECEF positions. Different standard deviations are considered for the random additive noise, based on the expected accuracy of different positioning techniques, as in the following table.

Positioning technique	**Error standard deviation σ**
SPP	1 m
PPP	1 dm
RTK	1 cm

TABLE 5.1: Different accuracy for the positioning techniques.

To test the attitude solver an error with a standard deviation of 1 dm has been initially used, which corresponds to the PPP noise. In figure 5.2 its probability density function is plotted.

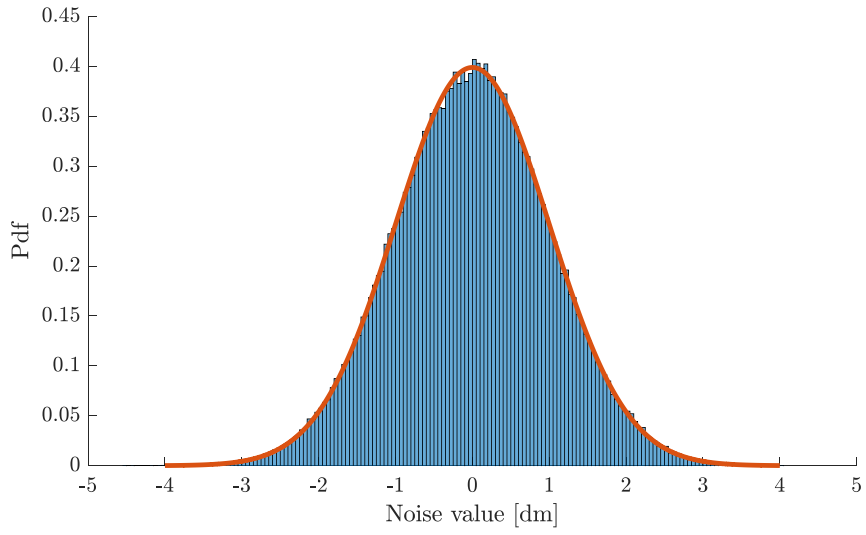

FIGURE 5.2: Pdf of the PPP noise used during simulation.

Then, the baselines in the ECEF frame are found by differentiating the antenna positions in the same frame. At this point, a rotation has to be

applied. Of course, being a simulation, the real rotations are known. In particular, to approximate a real movement of a vehicle over the time and to be sure that the results are good not only for a particular numerical value, the references angles are defined as sinusoidal functions, as shown in Figure 5.3.

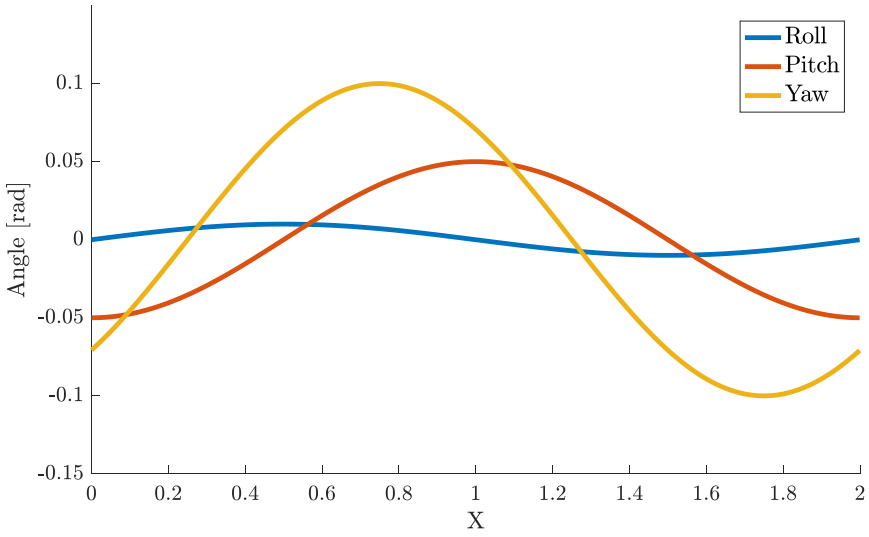

FIGURE 5.3: Definition of the reference Euler angles for each epoch.

Thus, for every epoch the reference angles for roll, pitch and yaw (or heading) are defined.

Both QUEST and Davenport have been applied, but since the differences are not significant, from now on, Davenport will be the algorithm realizing the attitude estimation. The results are showed in figure 5.4.

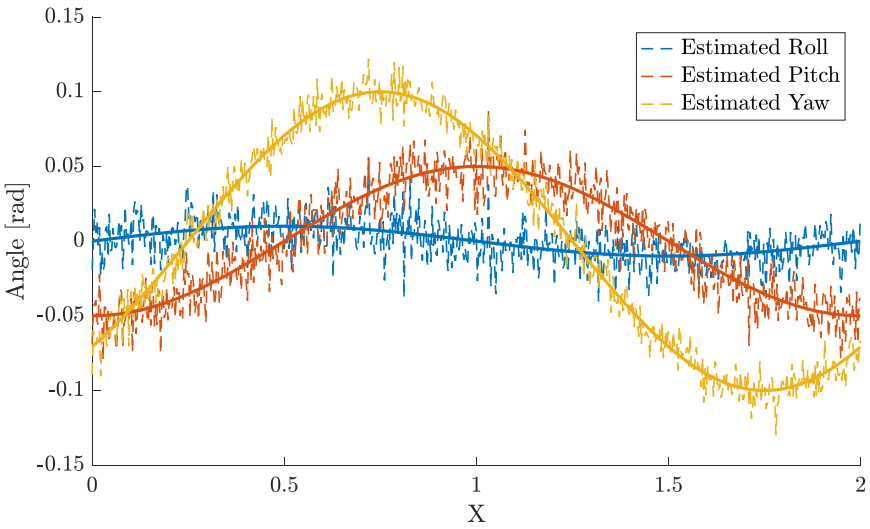

FIGURE 5.4: Estimated Euler angles.

The estimation is good for all the 3 angles and it perfectly follows the behaviour of the real ones.

By the way, it is very important to highlight that all the results are acquired through *Montecarlo simulations*. This is because a random noise is added. Therefore, it is important to have meaningful results from a statistical point of view. In dact, they do not have to be linked to a particular sample function (or realization) of the stochastic process representing the noise.

5.2 Factors affecting the attitude

Once has been verified that the attitude solver works properly, various simulation have been realized with the goal of understanding the impact of the different parameters on the attitude accuracy. In fact, as explained in Chapter 2, it will depend on:

- baseline length;
- accuracy of the positioning;
- geometry and number of the antennas.

However, before showing the next results is better to point out what is the meaning of the abscissa and ordinate axis of all the following plots. In fact, on the abscissa there are the distances between the antennas on the xy plane. This means that, the z coordinates are not changed. Thus, it is not really the baseline length, but still it has the same decreasing or increasing behavior. This is done with the purpose of allowing an easier view of the plots (the range of the values of the attitude error is lower). Finally, on the ordinate the mean value of the standard deviations of the 3 rotation angles is represented. In this way, it is much easier to see the impact of the different parameters on the overall attitude.

5.2.1 Varying baseline length

As showed at the end of Chapter 2, a geometrical approach can be used to check the behaviour of the attitude accuracy when the baseline length is changed. The model used is present in Figure 3.5, at page 22, and its mathematical representation, using the small angle assumption ($l >> \delta p$), is:

$$\delta\psi \simeq \frac{\delta p}{l} \tag{5.1}$$

where l represents the baseline length, δp the positioning error and $\delta\psi$ is the heading error. Since there is this model available, it is possible to check if the results obtained by the simulation are good and consistent, i.e. if the behavior is still descendant when the baseline length increases.

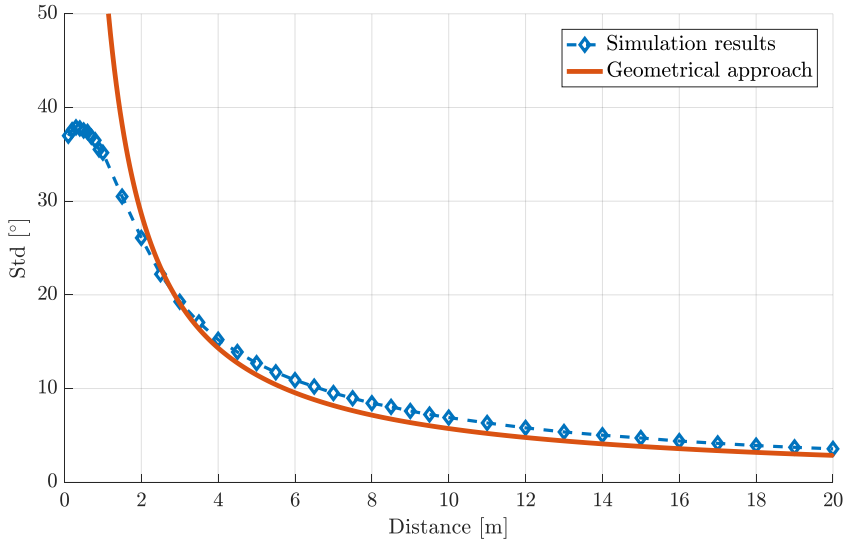

FIGURE 5.5: Comparison between the geometrical model and the simulation results, for an error with standard deviation of 1 m, varying the baseline length.

In figure 5.5 is showed the comparison between the simulation results, using the "triangular configuration" described in Figure 5.1, and the mathematical model when the SPP technique for the positioning is used ($\sigma = 1$ m). It is clear that the curves are different, especially for small baseline lengths, but this is not important. In fact, it depends on the very different assumptions between the simulation and the model, which are: 3 baselines instead of 1, estimation of the overall attitude error instead of the heading error and usage of the small angle assumption. However, the aim of this result is only to show that, also in the simulations, the error on the attitude decreases when the baseline length is increased. Another interesting aspect of the result in Figure 5.5 is that, for baseline lengths below 1 m, the behavior of the curve (for the simulation) is increasing. Anyway, this is due to the error on the positioning, which is comparable to the baseline length, and probably to a numerical error. In fact, this "strange" effect is not visible, or at least reduced, in case a smaller error is used, as is plotted in Figure 5.6). This is because the standard deviation of the error is now 10 cm (PPP error) and the smallest baseline length tested in the simulations is 0.5 m.

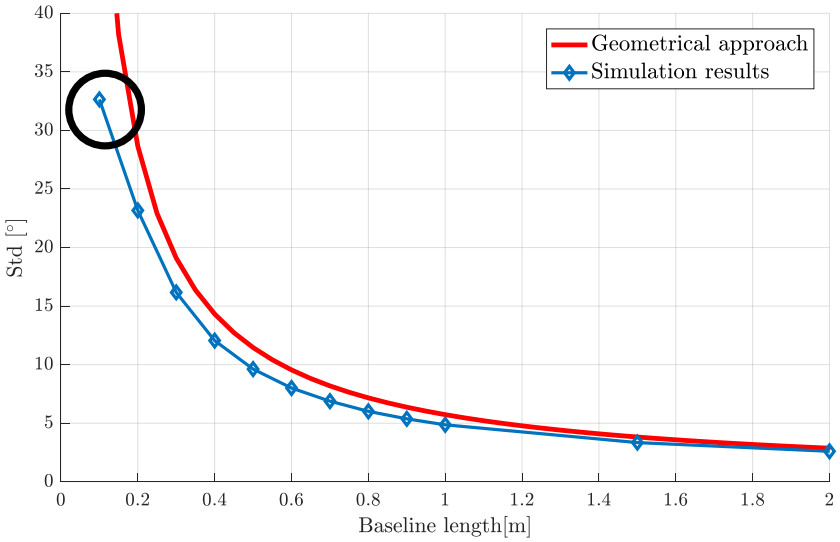

FIGURE 5.6: Comparison between the geometrical model and the simulation results, for a smaller error with a standard deviation of 10 cm, varying the baseline length.

5.2.2 Varying the accuracy

Another factor affecting the results can be the accuracy of the positioning technique. Using the configuration described at the beginning of this chapter, the results obtained from the simulation are showed in Figure 5.7. This is done for all the values of table 5.1.

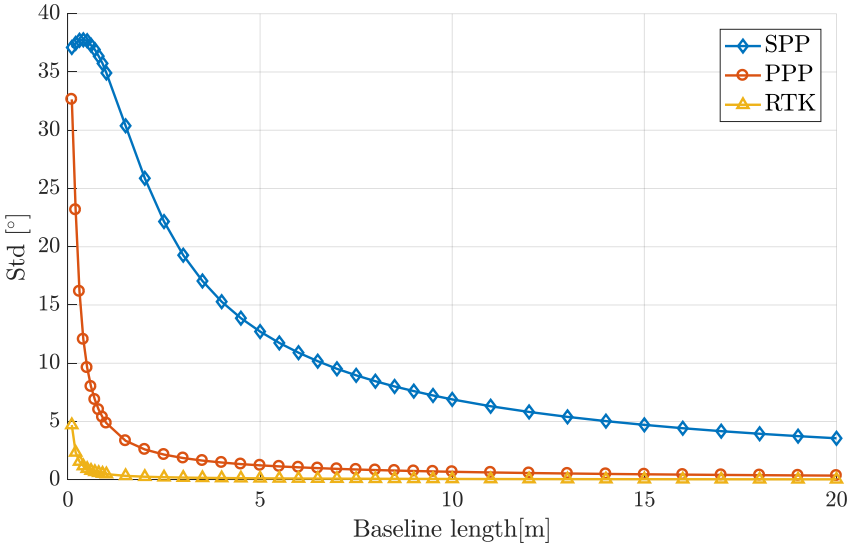

FIGURE 5.7: Standard deviation of the attitude error, for different accuracy of the positioning. See also table 5.1.

As expected, the error decreases when a better solution for the positioning is available. Furthermore, it is important to highlight that, when the SPP is used, the attitude error for a baseline length of 20 m is still larger than the one achieved for PPP when the baseline is long 2 m. Thus, it is clear that the accuracy has an higher impact on the results with respect to the baseline length.

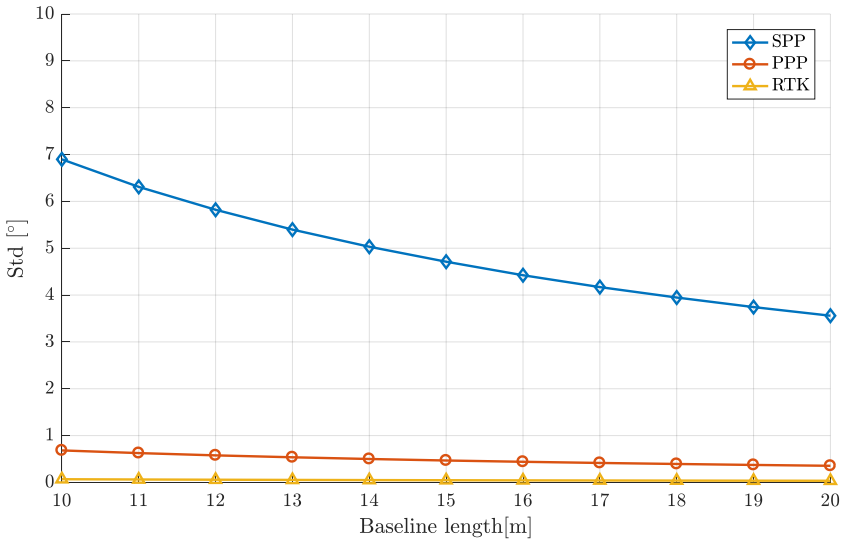

FIGURE 5.8: Zoom of the figure 5.7. In this figure it is easier to see the difference for large baseline lengths.

5.2.3 Varying the geometry

Again, as explained in Chapter 2, also the geometry can be relevant for the results. By the way, it is important to say that all the next results, for different geometries, are compared to the initial 3 antennas configuration of Figure 5.1, that will be called "Triangular configuration" or "Triangular geometry".
Moreover, all the curves, if not differently written, refers to the case of SPP accuracy. In practice, it is kind of a worst-case scenario analysis. In fact, with better accuracy, as proved previously, better results are expected.
First of all, to verify the importance of this parameter, a "bad" (from a theoretical point of view) configuration can be used, like the one in Figure 5.9, in which all the antennas are aligned over the x axis.

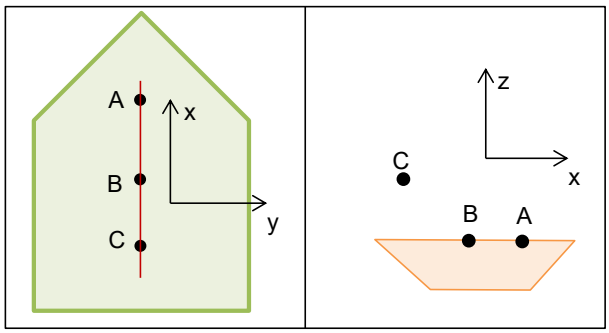

FIGURE 5.9: Bad configuration, with all the antennas aligned on the x axis.

Now, comparing the standard deviation of the attitude error achieved in this case with the one for the initial configuration of figure 5.1, a worse result is evident (see Figure 5.10).

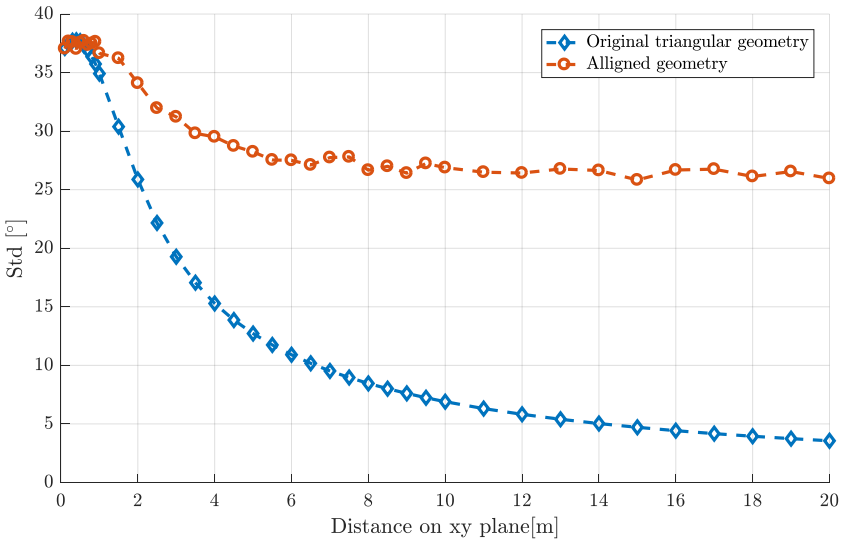

FIGURE 5.10: Comparison of the 2 different configurations for the antenna positions.

The error, for the "alligned" geometry, is never below the 25°. This is due to the fact that we are almost blind to x axis and, so, the roll angle cannot be well estimated. The overall performance is not acceptable.
Therefore, in every application the geometry has to been properly chosen. But, of course, the possible geometry depends on the shape of the rigid body.

However, changing the geometry means also changing the number of antennas. In fact, it is really interesting to understand what happens when one or more antennas are added to the configuration. For example, a geometry with 4 antennas, like the one in Figure 5.11 can be chosen. This kind of geometry can be called "Square geometry", because of its shape on the xy plane.

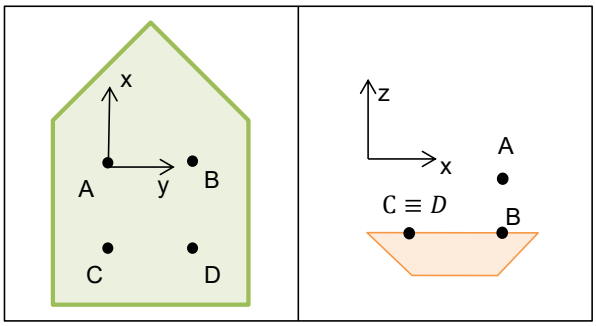

FIGURE 5.11: "Square geometry". 4 antennas system.

At the same time, another geometry with 4 antennas can be tested. This is done to avoid to have results bounded only to one particular configuration. For this reason the "Rhombus geometry" of Figure 5.12 is also evaluated.

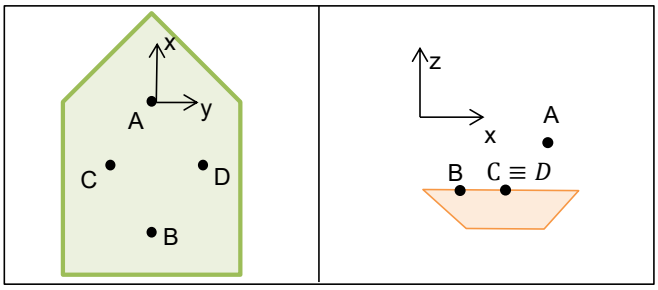

FIGURE 5.12: "Rhombus geometry". 4 antennas system, but with a different shape.

Through the same simulations as before, the results of figure 5.13 are obtained.

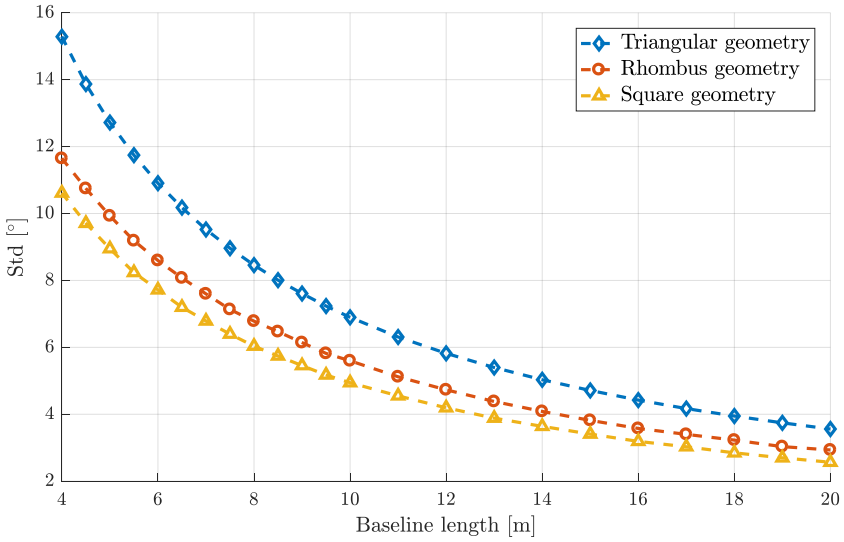

FIGURE 5.13: Comparison among the results attained using a different number of antennas (3 or 4).

This results clearly show that with 4 antennas (in both the "square" and the "rhombus" configurations) the estimation is better than the one using 3 antennas ("triangular configuration"). Thus, it is easy to understand that an higher number of well placed antennas allow better results.

Surely, the number of antennas can be still increased. For example, a "Penthagon configuration" with 5 antennas, like the one in Figure 5.14, can be used.

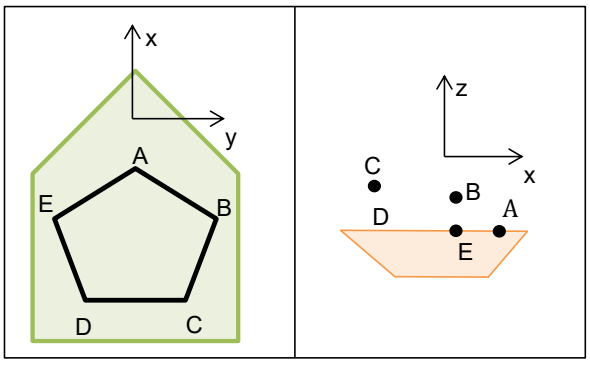

FIGURE 5.14: "Penthagon geometry". 5 antennas system.

Again, as showed in Figure 5.15, the results for this new configuration are better than the previous ones. However, an important trend is showed here. The impact of a new antenna on the final results is much smaller when the number of available antennas is larger. In fact, the distance between the

curve related to the "Penthagon" and the "Square" configurations, is lower than the one between the "Triangular" and "Square" ones.

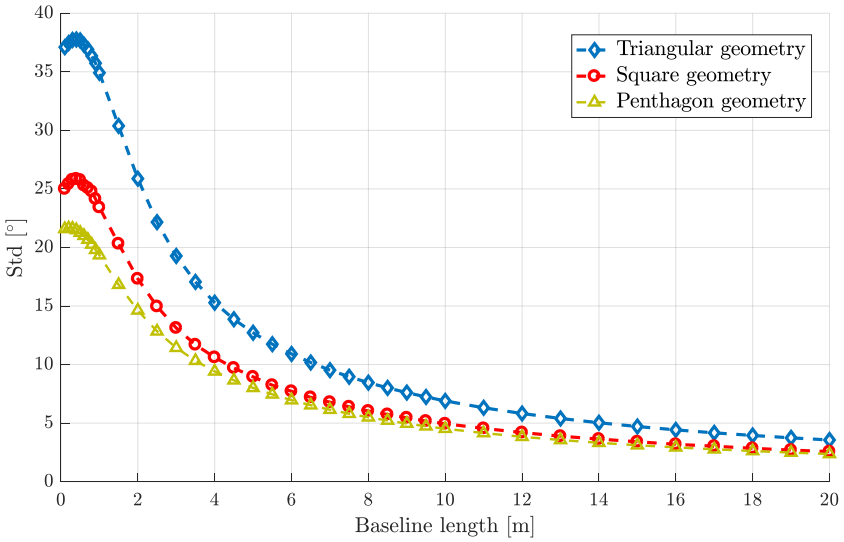

FIGURE 5.15: Comparison among the results attained using 3, 4 and 5 antenna systems.

5.3 Remarks

So, in these first simulations has been proved that, for the attitude estimation, is fundamental to take into account:

- accuracy of the positioning (higher accuracy means better attitude estimation);
- baseline length (the accuracy improves for higher baseline length);
- geometry of the antennas (as well as their number).

Finally, an important remark has to be state after seeing the previous results. Indeed, having many antennas does not always mean huge improvements in the accuracy. When the accuracy of the positioning is good, there is very small gain and this is not worth compared to the cost. Thus, the usage of many antennas could be justified only in the case of the relative positioning of 2 moving platforms.
At the same time, the number of antennas is bounded to the size of the vehicle. Indeed, having many antennas, on a very small vehicle would mean to have many short baselines, i.e. there would not be any improvement. For example, having 5 antennas on a car would be impossible.

It is clear that the positioning accuracy is the most relevant factor. Thus, the use of GNSS phase observations is justified.

5.4 GNSS attitude simulation

In this last section, differently to the previous ones, a simulation related to the Chapter 4 is analyzed.
The body and reference frame of Figure 5.16 is taken into account. It means that there are 3 antennas and, consequently, 3 baselines.

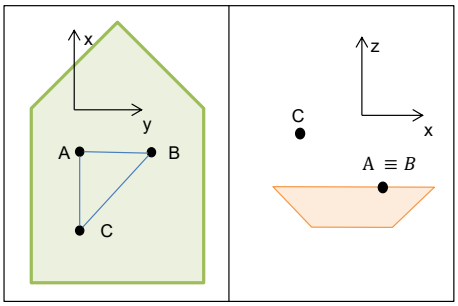

FIGURE 5.16: Body frame and system used for the simulation.

Then, a reference rotation is chosen. This value does not affect the results. Thus, a simple rotation of values $[\varphi, \theta, \psi] = [0°, 0°, 90°]$ can be applied to our body frame with respect to the ECEF one. But, in order to simulate a real scenario behavior, some random noise has to be added to it. It is really important to point out that a non recursive simulation is done. In other words, the results are focused on a *single epoch*, to which corresponds the sky plot of Figure 5.17. It is understandable that there are 10 satellites in view.

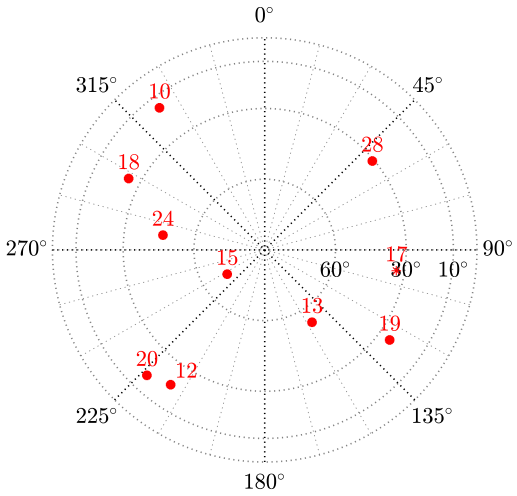

FIGURE 5.17: Sky plot of the satellites in view for the single epoch analyzed in the simulation.

Besides, the code and phase measurements, as well as, the ambiguities are fixed for that epoch.
At last, the goal of the following simulation is to check what is the effect of the:

- baseline length;
- PDOP (it can be changed varying the number of satellites available);
- accuracy of the code measurements (the one of the phase observations is kept fixed to 3 mm);

on the fixing ratio and the success rate, for both the "Classical approach" and the "Quaternion approach", and to compare their performance.
In table 5.2 all the necessary parameters are displayed. All the possible combination have been simulated through Montecarlo simulations, in order to have meaningful outcomes.

TABLE 5.2: The different simulated measurement scenarios.

UTC time and date	Date XXX
Frequency	L1
Number of satellites	Corresponding PDOP
5 / 6 / 7 / 8 / 9	6.413 / 2.403 / 2.041 / 1.988 / 1.811
Code noise σ_R [cm]	30 - 15 - 5
Phase noise σ_Φ [mm]	3
Baseline length [m]	0.5 - 2 - 5 - 10 - 20 - 50 - 100
Samples simulated	10^4

The outcomes of the simulations, for both the approaches, are present in Figure 5.18. These are the results for an accuracy of $\sigma_\rho = 30cm$ and for a number of satellites that goes from 5 to 10. What is plotted is, then, the percentage of the success ratio, i.e. the number of times in which the estimated ambiguity vector is equal to the nominal one, when the baseline length is increased.

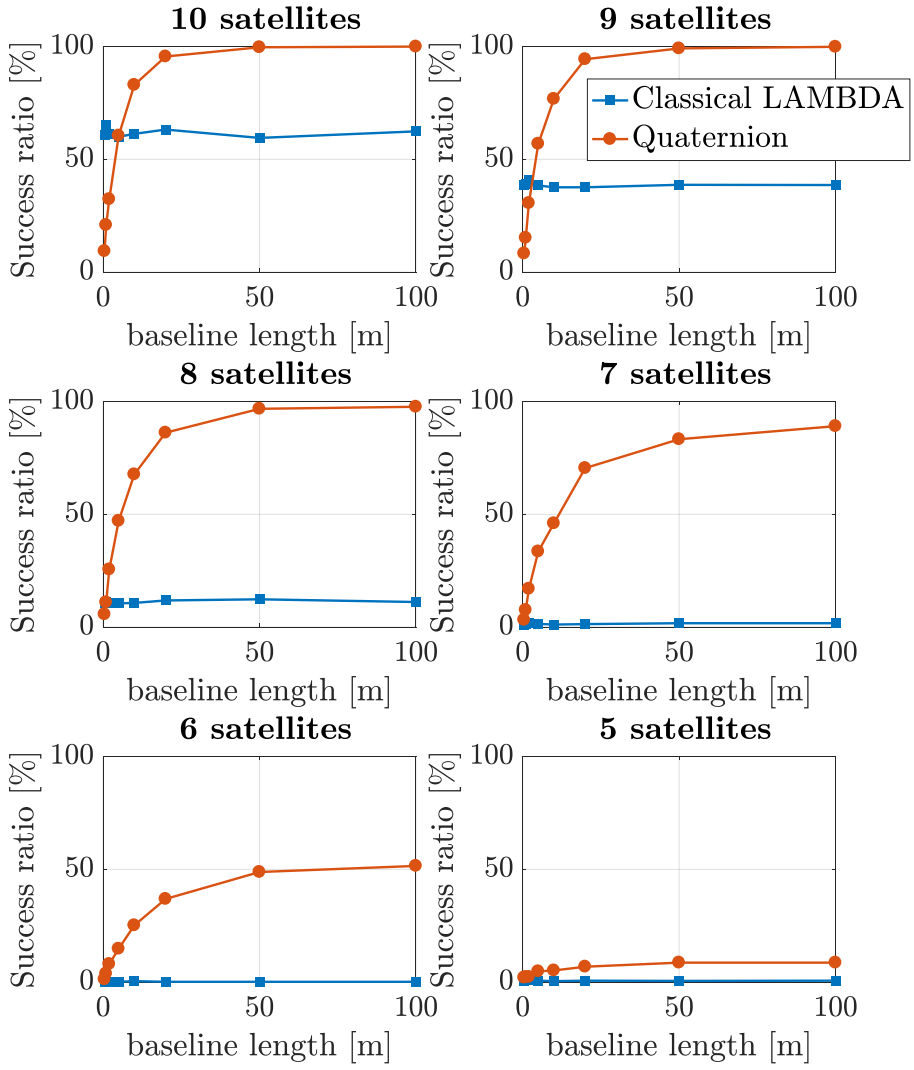

FIGURE 5.18: Success ratio vs baseline length for $n = 5$ to $n = 10$ satellites in view, code accuracy $\sigma = 30cm$, for both the classical LAMBDA method (orange) and the quaternion based(blue).

It is clearly visible that the success ratio increases together with the number of satellites in view, i.e. when a better PDOP is achieved. For example, for the classical LAMBDA method, the ratio goes from 0%, in case of 5 satellites, up to 63%. For what matters the Quaternion based approach the increment, for very long baselines ($100m$), is also better: from 8% to 100%. But,

for short baselines this is not true anymore.
Here it comes one of the biggest differences between these methods. The new proposed one is, as a matter of fact, dependent from the baseline length. In particular, when the distance between the antennas is larger the success rate increases, as well. The explanation for this trend, has to be found in the model of Chapter 4. In fact, it depends on the matrix $F_q^i = \frac{\partial(q \odot b_i \odot q^*)}{\partial q}$, which creates a dependence on the baselines. It is also interesting to notice that the curve has a really high slope up to a $20m$ length. On the contrary, the success ratio for the classical approach remains constant over the length. This means that for some applications this method could give better performance, as is visible in the case of 10 satellites in Figure 5.18. In this situation, indeed, for baselines shorter than $5m$ the results for the quaternion based approach start to be worse.
It is also interesting to see what is the effect of the code accuracy on the results. An accuracy $\sigma = 15cm$ has been tested and the outcome is the one in Figure 5.19.

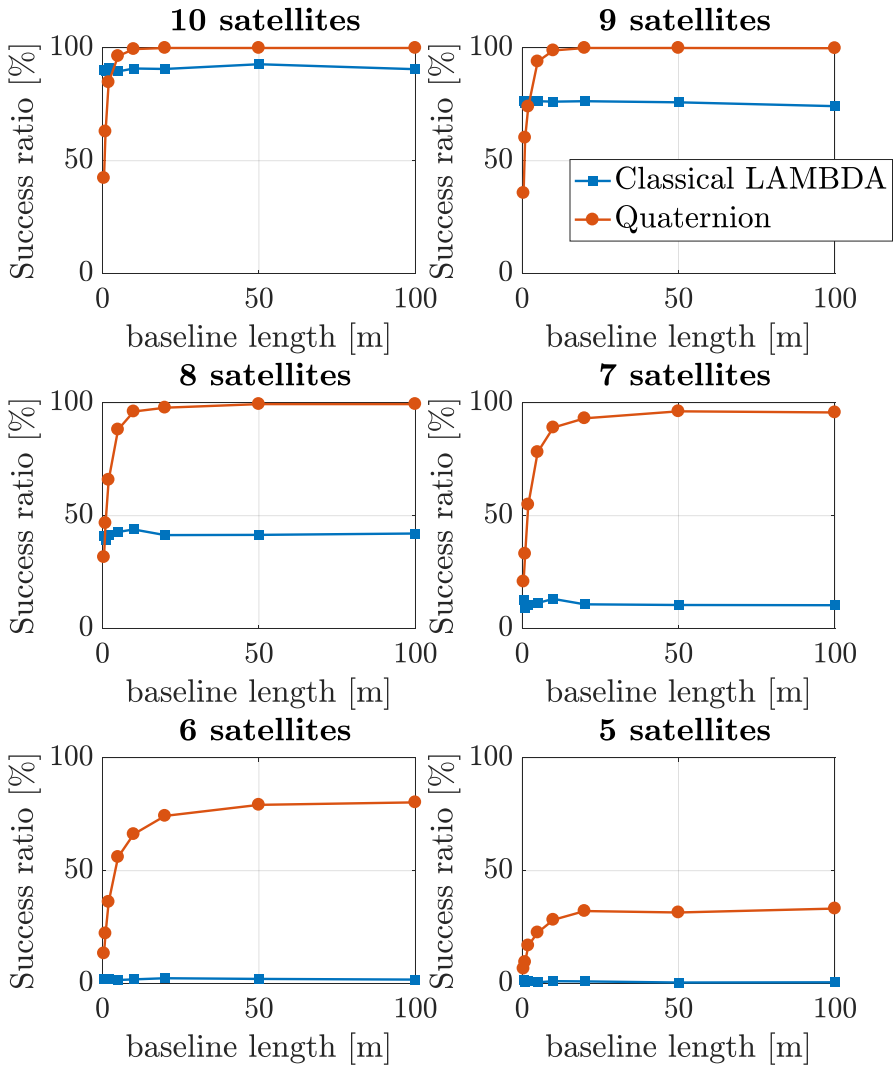

FIGURE 5.19: Success ratio vs baseline length for $n = 5$ to $n = 10$ satellites in view, code accuracy $\sigma = 15cm$, for both the classical LAMBDA method (orange) and the quaternion based(blue).

The behaviour with respect to the baseline length is always the same, i.e. it grows, but better success ratio are achieved, as expected. The best improvements, however, are obtained fir the quaternion based approach. In fact, now, it is always better than the classical one for baselines longer than $3m$.

Some remarks can now be done. The quaternion based gives usually better results, especially in worse conditions, i.e. for a bad PDOP. However, due to its dependence on the baseline length l, at some point the success ratio for the classical LAMBDA starts to be better. This crossing point depends on the simulation parameters. This means that the choice of the best method to be used depends on the specific application.
Moreover, with these results is also proved that the linearization, using the line of sight vectors, is a good approximation. In fact, the accuracy for the attitude is very good. Checking, the standard deviation of the error, it is always below 0.5° for the quternion based, while it is usually below 1° for the classical method.

It is also interesting what happens to the fixing ratio. Its behavior is showed in Figure 5.20. To decide if the solution is fixed or not a threshold of value $\gamma = 3$ is used.

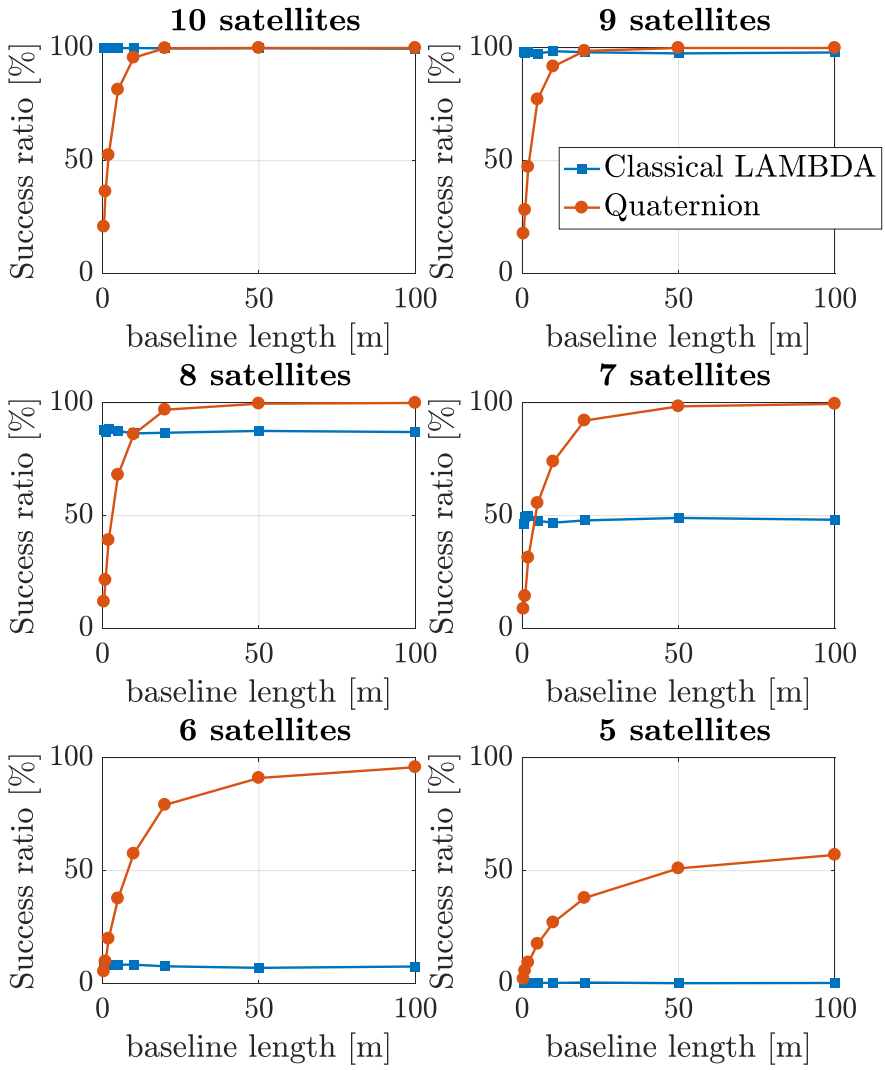

FIGURE 5.20: Fixing ratio vs baseline length for $n = 5$ to $n = 10$ satellites in view, code accuracy $\sigma = 30cm$, for both the classical LAMBDA method (orange) and the quaternion based(blue).

It is clear that the value chosen as threshold is not the optimal one for this simulation. By the way, to set it properly a statistical analysis, based on the target false alarm probability, should be performed. In addition, it is really interesting to observe that the threshold should not be the same for the two methods. In fact, for the classical LAMBDA approach the chosen value is

too optimistic, because the fix ratio is much larger than the success ratio. On the contrary, for the quaternion based method, the threshold seems to be close to the optimal value. Indeed, the fix ratio is accurate enough.

Conclusion

To sum up, in this thesis, the attitude problem for GNSS has been faced. First of all, it has been showed how some parameters affects the results and the accuracy of the estimated Euler angles. In particular, it has been showed the importance of using phase measurements.
After that, a new approach for finding a better estimation has been proposed, i.e. the *Quaternion approach*. It shows better results, in terms of success ratio, with respect to the classical LAMBDA method, but not for short baselines.
Still, many steps have to been done in this research topic. In particular, a comparison with the MC-LAMBDA method should be done. In fact, MC-LAMBDA (Multivariate Constrained LAMBDA) [5] [6] is an evolution of the LAMBDA method and it differs for the addition of a new constraint. Indeed, the baseline length in the body frame are known and this information can be used to get an improvement of the performance in the MC-LAMBDA. Moreover, all these methods should be studied and analysed into an actual dynamical scenario. In fact, this behaviour and the performance should be validated using real GNSS data, because, in real life, better or worse results could be obtained.

Bibliography

[1] Itzhack Y Bar-Itzhack and Richard R Harman. "Optimized TRIAD algorithm for attitude determination". In: *Journal of guidance, control, and dynamics* 20.1 (1997), pp. 208–211.

[2] Arias Medina Daniel. *Computational Aspects of Sensor Fusion for GNSS Outlier Mitigation in Navigation*. UNIVERSIDAD CARLOS III DE MADRID, 2016.

[3] James Diebel. "Representing attitude: Euler angles, unit quaternions, and rotation vectors". In: *Matrix* 58.15-16 (2006), pp. 1–35.

[4] Yanming Feng and Jinling Wang. "GPS RTK performance characteristics and analysis". In: *Journal of Global Positioning Systems* 7.1 (2008), pp. 1–8.

[5] Gabriele Giorgi and Peter Teunissen. "GNSS carrier phase-based attitude determination". In: *Recent advances in aircraft technology*. InTech, 2012, pp. 193–220.

[6] Gabriele Giorgi and Peter JG Teunissen. "Carrier phase GNSS attitude determination with the multivariate constrained LAMBDA method". In: *2010 IEEE Aerospace Conference*. IEEE. 2010, pp. 1–12.

[7] Gabriele Giorgi et al. "Testing a new multivariate GNSS carrier phase attitude determination method for remote sensing platforms". In: *Advances in Space Research* 46.2 (2010), pp. 118–129.

[8] *Google Maps*. http://maps.google.com/. 2019.

[9] Paul D Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech house, 2013.

[10] J.M. Juan Zornoza J. Sanz Subirana and M. Hernández-Pajares. *GNSS Data Processing, Vol. I: Fundamentals and Algorithms. Vol. I: Fundamentals and Algorithms*. ESA Communications, 2013.

[11] W. Jiang, Y. Li, and C. Rizos. "Optimal Data Fusion Algorithm for Navigation Using Triple Integration of PPP-GNSS, INS, and Terrestrial Ranging System". In: *IEEE Sensors Journal* 15.10 (2015), pp. 5634–5644. ISSN: 1530-437X. DOI: 10.1109/JSEN.2015.2447015.

[12] Elliott Kaplan and Christopher Hegarty. *Understanding GPS: principles and applications*. Artech house, 2005.

[13] Anthony Kim and MF Golnaraghi. "A quaternion-based orientation estimation algorithm using an inertial measurement unit". In: *PLANS 2004. Position Location and Navigation Symposium (IEEE Cat. No. 04CH37556)*. IEEE. 2004, pp. 268–272.

[14] Ern J Lefferts, F Landis Markley, and Malcolm D Shuster. "Kalman filtering for spacecraft attitude estimation". In: *Journal of Guidance, Control, and Dynamics* 5.5 (1982), pp. 417–429.

[15] Carl Ch Liebe. "Star trackers for attitude determination". In: *IEEE Aerospace and Electronic Systems Magazine* 10.6 (1995), pp. 10–16.

[16] F Landis Markley. "Attitude determination using vector observations: A fast optimal matrix algorithm". In: (1993).

[17] F Landis Markley. "Attitude determination using vector observations and the singular value decomposition". In: *The Journal of the Astronautical Sciences* 36.3 (1988), pp. 245–258.

[18] F. Landis Markley and Daniele Mortari. "How to Estimate Attitude from Vector Observations". In: *AAS Paper 99-427, presented at the AAS/AIAA Astrodynamics Specialist Conference, Girdwood, Alaska* (1999).

[19] Daniel Medina et al. "On the Kalman filtering formulation for RTK joint positioning and attitude quaternion determination". In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE. 2018, pp. 597–604.

[20] Pratap Misra and Per Enge. "Global Positioning System: signals, measurements and performance second edition". In: *Massachusetts: Ganga-Jamuna Press* (2006).

[21] Daniele Mortari. "ESOQ: A closed-form solution to the Wahba problem". In: *Journal of the Astronautical Sciences* 45.2 (1997), pp. 195–204.

[22] *RTK Fundamentals*. 2011. URL: https://gssc.esa.int/navipedia/index.php/RTK_Fundamentals.

[23] Simone Sabatelli et al. "A double stage Kalman filter for sensor fusion and orientation tracking in 9D IMU". In: *2012 IEEE Sensors Applications Symposium Proceedings*. IEEE. 2012, pp. 1–5.

[24] Chalermchon Satirapod, Chris Rizos, and Jinling Wang. "GPS SINGLE POINT POSITIONING WITH SA OFF: HOW ACCURATE CAN WE GET?" In: *Survey Review* 36.282 (2001), pp. 255–262. DOI: 10.1179/sre.2001.36.282.255. eprint: https://doi.org/10.1179/sre.2001.36.282.255. URL: https://doi.org/10.1179/sre.2001.36.282.255.

[25] M Shuster. "Approximate algorithms for fast optimal attitude computation". In: *Guidance and Control Conference*. 1978, p. 1249.

[26] Malcolm D Shuster. "The quest for better attitudes". In: *The Journal of the Astronautical Sciences* 54.3-4 (2006), pp. 657–683.

[27] Joan Sola. "Quaternion kinematics for the error-state KF". In: *Laboratoire dAnalyse et dArchitecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep* (2012).

[28] P. J. G. Teunissen. "Success probability of integer GPS ambiguity rounding and bootstrapping". In: *Journal of Geodesy* 72.10 (1998), pp. 606–612. ISSN: 1432-1394. DOI: 10.1007/s001900050199. URL: https://doi.org/10.1007/s001900050199.

[29] Peter Teunissen. "A general multivariate formulation of the multi-antenna GNSS attitude determination problem". In: *Artificial Satellites* 42.2 (2007), pp. 97–111.

[30] Peter Teunissen. "The LAMBDA method for the GNSS compass". In: *Artificial Satellites* 41.3 (2006), pp. 89–103.

[31] Peter Teunissen and Oliver Montenbruck. *Springer handbook of global navigation satellite systems*. Springer, 2017.

[32] Peter JG Teunissen. "An optimality property of the integer least-squares estimator". In: *Journal of geodesy* 73.11 (1999), pp. 587–593.

[33] P.J.G. Teunissen. "Least-Squares Estimation of the Integer GPS Ambiguities". In: *Delft Geodetic Computing Centre (LGR)* (1993).

[34] *EarthNow promises real-time views of the whole planet from a new satellite constellation, Devin Coldewey*. URL: https://tcrn.ch/2vqMz51.

[35] *MISB Standard 0601 - defining much of aerospace vocabulary as well as georeferencing standard for videos*. URL: https://en.wikipedia.org/wiki/File:MISB_ST_0601.8_-_Yaw,_Pitch_%26_Roll.png.

[36] S Verhagen, B Li, and Mathematical Geodesy. "LAMBDA software package: Matlab implementation, Version 3.0". In: *Delft University of Technology and Curtin University, Perth, Australia* (2012).

[37] James R Wertz. *Spacecraft attitude determination and control*. Vol. 73. Springer Science & Business Media, 2012.